Adaptive Inference on General Graphical Models

Umut A. Acar[*]
Toyota Tech. Inst.
Chicago, IL
umut@tti-c.org

Alexander T. Ihler
U.C. Irvine
Irvine, CA
ihler@ics.uci.edu

Ramgopal R. Mettu[†]
Univ. of Massachusetts
Amherst, MA
mettu@ecs.umass.edu

Özgür Sümer
Univ. of Chicago
Chicago, IL
osumer@cs.uchicago.edu

Abstract

Many algorithms and applications involve repeatedly solving variations of the same inference problem; for example we may want to introduce new evidence to the model or perform updates to conditional dependencies. The goal of *adaptive inference* is to take advantage of what is preserved in the model and perform inference more rapidly than from scratch. In this paper, we describe techniques for adaptive inference on general graphs that support marginal computation and updates to the conditional probabilities and dependencies in logarithmic time. We give experimental results for an implementation of our algorithm, and demonstrate its potential performance benefit in the study of protein structure.

1 Introduction

It is common in many applications to repeatedly perform inference on a variations of essentially the same graphical model. For example, in a number of learning problems we may use observed data to modify a portion of the model (e.g., fitting an observed marginal distribution), and then recompute various moments of the new model before updating the model further [8]. Another example is in the study of protein structures, where a graphical model can be used to represent the conformation space of a protein structure [15, 9]. The maximum-likelihood configuration in this model then corresponds to the minimum-energy conformation for the corresponding protein. An application of interest in this setting is to perform amino acid mutations in the protein to determine the effect of these mutations to the structure and the function of the protein.

The changes described in the examples above can, of course, be handled by incorporating them into the model and then performing inference from scratch. However, in general we may wish to assess thousands of potential changes to the model; for example, the number of possible mutations in a protein structure grows exponentially with the number of considered sites. *Adaptive inference* refers to the problem of handling changes to the model (e.g. to conditional dependencies and even graph structure) more efficiently than performing inference from scratch. Delcher *et al.* [6] studied this problem under a set of fairly restrictive conditions, requiring that the graph be tree-structured and supporting only changes to the observed evidence in the model. They show that updates to observed evidence may be performed in expected $O(\log n)$ time, where n is the size of the graph. More recently, Acar *et al.* [2] gave a method of supporting more general changes to the model so long as the model remains tree-structured.

Unfortunately, many graphical models of interest are not trees, but are "loopy". In principle, we can perform adaptive inference on loopy graphs by construcing their junction tree [13] and applying existing frameworks to the junction tree itself [6, 2]. This approach, however, can be very slow since even a small change to the graph can cause the junction tree to change dramatically, e.g., creating a cycle by inserting a new edge can require a linear number of changes to the junction tree.

In this paper, we present techniques for supporting adaptive inference on general graphical models efficiently. Given a factor graph G with n nodes, maximum degree k, and domain size d (variables can take d different values), we require the user to specify a spanning tree T of G. We then construct a *(hierarchical) clustering* of G with respect to the spanning tree T (Sec. 3). The hierarchical clustering is a tree of clusters where each cluster represents a subgraph of G. A key property of the clustering is that it has expected $O(\log n)$ depth, where the expectations are taken over internal randomization. For each cluster we compute a *cluster function*, a partial marginalization of factors in the cluster. We show that the cluster functions can be computed in $O(\alpha^m)$ where $\alpha = d^{k+1}$ and m is the size of the boundary of the cluster.

[*]U. A. Acar is supported by a gift from Intel.

[†]R. R. Mettu is supported by a National Science Foundation CAREER Award (IIS-0643768).

Given such a hierarchical clustering, we show how to compute the marginal at any variable by performing a traversal from the top level cluster to the variable. Since maximum path length in the clustering is expected $O(\log n)$, we show that marginals can be computed in expected $O(\alpha^\beta \log n)$ time where β is an upper bound on the boundary size of all clusters (for a tree-structured factor graph $\beta = 2$). The novel contribution of our approach is that our clustering also allows efficient updates to factors and edge insertions/deletions in the input graph. We show that after any of these updates is applied, it is possible to update the clustering $O(\alpha^\beta \log n)$ time and that marginals computed thereafter correctly reflect the updates.

Our results generalize the previous techniques for adaptive inference with tree-structured factor graph to loopy graphs. The main insight is to partition the loopy graph into a spanning tree and a set of non-tree edges and cluster the graph based on the spanning tree only. This enables updating the hierarchical clustering in expected logarithmic time when an edge is inserted or deleted using RC-Trees. When computing marginals, contributions of the nodes of the graph are computed in the order specified by the clustering on the spanning-tree edges. Compared to previous work on factor trees [2], we also simplify marginal computations.

We note that our bounds depend exponentially on the boundary size of the clusters. While this exponential cost can be large in general, for many interesting classes of graphs it can be kept small. Moreover, since our expected running times are logarithmic in n, our approach can still be significantly faster than computing from scratch. This exponential factor is not surprising, since exact inference on general graphs is NP-hard; conventional, algorithms for exact inference also have an exponential dependence on some property of the graph such as the tree width.

To evaluate the effectiveness of the proposed techniques, we implemented our algorithm and compared its performance against an implementation of sum-product that performs inference on a junction-tree of the given factor graph. Our experiments on a synthetic benchmark for factor graphs show that our approach can be orders of magnitude faster than sum-product. We also investigate the applicability of our algorithm to study protein structure, and show that our algorithm is considerable faster than sum-product for modeling several moderately-sized proteins.

2 Background

Graphical models provide a convenient formalism for describing structure within a function $g(X)$ defined over a set of variables $X = [x_1, \ldots, x_n]$ (most commonly a joint probability distribution or energy function over the x_i). Graphical models use this structure to organize computations involving $g(\cdot)$ and construct efficient algorithms for many inference tasks, including optimization to find a maximum a posteriori (MAP) configuration, marginalizing, or computing the likelihood of observed data. For the purposes of this paper, we assume that each variable x_i takes on values from some finite set and focus primarily on the problem of marginalization.

2.1 Factor Graphs

Factor graphs [10] describe the factorization structure of the function $g(X)$ using a bipartite graph consisting of *factor* nodes and *variable* nodes. Specifically, suppose such a graph G consists of factor nodes $F = \{f_1, \ldots, f_m\}$ and variable nodes $X = \{x_1, \ldots, x_n\}$, and let $X_j \subseteq X$ denote the neighbors of factor node f_j. Then, G is said to be consistent with a function $g(\cdot)$ if and only if

$$g(x_1, \ldots, x_n) = \prod_j f_j(X_j).$$

In a common abuse of notation, we have used the same symbols to indicate both each variable node and its associated variable x_i, and similarly for each factor node and its associated function f_j.

It will often be convenient to refer to vertices without specifying whether they are variable or factor nodes. To this end, we define a set of artificial "factors" to be associated with both factors and variable nodes; for a generic vertex v we define $\psi_v(X_v) \equiv 1$ for $v = x_i$, and $\psi_v(X_v) = f_j(X_j)$ for $v = f_j$.

2.2 Marginalization

A classic inference problem is that of marginalizing the function $g(X)$. Specifically, for some or all of the x_i, we are interested in computing the marginal function

$$g^i(x_i) = \sum_{X \setminus x_i} g(X).$$

When the factor graph representation of $g(X)$ is singly-connected (tree-structured), marginalization can be performed efficiently using sum-product [10]. In tree-structured graphs, sum-product is typically formulated as a two-pass sequence: rooting the tree at some node v, messages are sent upward (leaves to root), then back downward, after which one may compute the marginal for any node in the graph. In more general graphs (graphs with cycles), exact inference is less straightforward. One solution is to use a *junction tree* [11]; this first constructs a tree-structured hypergraph of G, then runs essentially the same inference process to compute marginals. The computational complexity of this process depends on the selected hypergraph and is exponential in the size of the cliques, or nodes of the hypergraph.

An alternate but essentially equivalent view of exact inference is given by the *bucket elimination* algorithm [5].

Bucket elimination chooses a sequence in which to marginalize the variables x_i, first multiplying together each of the factors which include x_i, then summing over x_i to create a new factor and returning it to the pool. In tree-structured graphs, a marginal function $g^i(x_i)$ can be found in a manner similar to the upward pass of sum-product: rooting the tree at the node x_i of interest, the summation operations are carried out first on the leaf nodes, followed by their parents, and so on until only the root x_i remains. However, bucket elimination does not impose any particular elimination order, and we shall see in the sequel that alternative orders may come with other benefits.

Bucket elimination is closely related to junction tree based inference, and an equivalent junction tree may be defined implicitly by its specified elimination ordering [5].

2.3 RC-Trees for Adaptive Inference

In [2], an algorithm for adaptive inference in factor trees is described using "rake and compress" trees (RC-trees). The RC-tree data structure automatically selects an elimination ordering for the variables in the factor tree using a random-mate selection procedure, and stores functions at each node in the RC-tree representing sufficient statistics for its subtree. It was shown that construction of the RC-tree data structure requires time and space linear in the number of vertices n of the factor graph, and produces a balanced tree with expected height $O(\log n)$.

The sufficient statistics stored in the RC-tree can be used to "query", or compute marginal distributions in the factor tree by passing information downward, taking at most expected $O(kd^{k+2}\log n)$ time, where k is the maximal degree of the factor tree, and d is the maximal dimension of each variable. Moreover, *changes* to the tree can also be incorporated in expected $O(kd^{k+2}\log n)$ time, including changes to the tree structure. The nature of the random-mate elimination ordering ensures that such changes affect only logarithmically many of the sufficient statistics.

Unfortunately, this formulation is restricted to tree-structured factor graphs, which limits its applicability in practice. In the following sections, we describe a generalization of the RC-tree structure which can cope with cycles in the factor graph while maintaining the desireable properties of the automatically chosen elimination ordering.

3 Hierarchical Clustering and Inference

We begin by describing a notion of hierarchical clustering in factor graphs which is compatible with but more general than that induced by RC-trees. We then describe how this clustering can be used to compute the marginal distribution at any vertex of the factor graph.

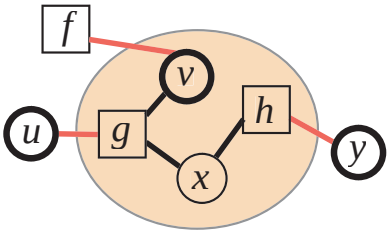


Figure 1: A cluster C (shaded) with boundary edges (red) $\partial C = \{(u,g),(v,f),(y,h)\}$, boundary variables (bold circles) $X_c = \{u,v,y\}$ and cluster function $\varphi_C = \sum_{X\setminus X_c} g\cdot h = \sum_x g(u,v,x)\cdot h(x,y)$.

3.1 Hierarchical Clustering

For a factor graph $G = (X+F, E)$, a *cluster* C is simply a set of vertices of G. We define the *boundary* of a cluster, written ∂C, as a set of edges with exactly one endpoint in C, and the *boundary variables* X_C of C to be the set of variables (variable nodes) incident to the boundary edges. For each cluster, we also define a *cluster function* φ_C as the partial marginalization of all the factors in that cluster over all variables except the boundary variables:

$$\varphi_C(X_C) = \sum_{X\setminus X_C}\prod_{f_j\in C} f_j(X_j).$$

Fig. 1 shows an example cluster, its boundary and boundary variables.

We can then define a *hierarchical clustering* of G to be a set of clusters $\mathcal{C} = C_1,\ldots,C_n$ such that the following conditions are satisfied:

1. Every vertex is covered by at least one cluster.
2. Clusters are nested: given two clusters either one is a subset of the other or they do not intersect. Moreover, if two clusters share a boundary edge, one is a subset of the other.
3. Each cluster C has a unique identifier vertex v : for any $C\in\mathcal{C}$ there is a unique $v\in C$ such that no other cluster contained by C contains v. We write $\bar{v}$ to denote the cluster of identified with vertex v, i.e., $\bar{v} = C$.
4. For each maximal subcluster C' of $C = \bar{v}$, i.e., C' contained in no smaller cluster than C, there is an edge connecting v and some $u\in C'$.

Fig. 2 shows a factor graph and a valid hierarchical clustering of the graph. Note that, by condition 3, the finest scale of the clustering are individual nodes.

A hierarchical clustering can be constructed bottom-up, by combining groups of sub-clusters which are adjacent to the same vertex. Since clusters are nested, we can represent a hierarchical clustering as a *cluster tree*, so that if a cluster C' is a subset of C, then C is an ancestor of C' in the tree;

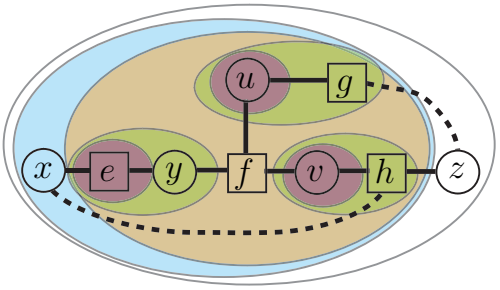


Figure 2: A factor graph G and hierarchical clustering of G. Edges of G designated as "non-tree" (see text) are shown as dashed.

the maximal subclusters of C are the children of C. A cluster tree representation of the clustering in Fig. 2 is shown in Fig. 3. In the cluster-tree, each cluster is labeled based on its identifier vertex, e.g., the cluster $\bar{u}$ has identifier u. Also shown for each cluster are the boundary edges.

The cluster boundaries and their cluster functions can be computed in the cluster tree recursively, based on those of their immediate children. Let $S_{\bar{u}} = \{\bar{v}_1, \ldots, \bar{v}_k\}$ be the set of children of $\bar{u}$ in the cluster tree, and let $E(u)$ denote the edges containing u as an endpoint. Then, the boundary of $\bar{u}$ is the set of edges that are in exactly one of $E(u), \partial\bar{v}_1, \ldots, \partial\bar{v}_k$, i.e.

$$\partial\bar{u} = E(u) \triangle \partial\bar{v}_1 \triangle \ldots \triangle \partial\bar{v}_k$$

where $\triangle$ is the symmetric set difference operator.

The cluster function for $\bar{u}$ can be computed as

$$\varphi_{\bar{u}}(X_{\bar{u}}) = \sum_{X \setminus X_{\bar{u}}} \psi_u(X_u) \prod_{\bar{v} \in S_{\bar{u}}} \varphi_{\bar{v}}(X_{\bar{v}}).$$

(Recall that the ψ_u simply refer to factors of $g(\cdot)$.) Any such hierarchical clustering can be used to define a (partial) elimination ordering, with a variable being eliminated in the first (bottom-most) cluster which contains both the variable and all its neighboring factors. In the bucket elimination algorithm following this partial ordering, each cluster function $\varphi_C(X_C)$ then corresponds to the "new factor" created by marginalizing the factors in a given bucket.

Finally, we will find it useful to partition the edges of G into two sets. In a hierarchical clustering $\mathcal{C}$, at each cluster $C = \bar{v}$ there exists at least one edge from $\bar{v}$ to each of its maximal subclusters C' (if there is more than one, we can break ties arbitrarily). The collection of these edges form a subtree (or forest) of the original factor graph. We call these edges the "tree" edges $E_T \subseteq E$ of the hierarchical clustering; the remaining edges $E_N = E \setminus E_T$ we call the "non-tree" edges. In Fig. 2, the non-tree edges E_N are shown as dashed.

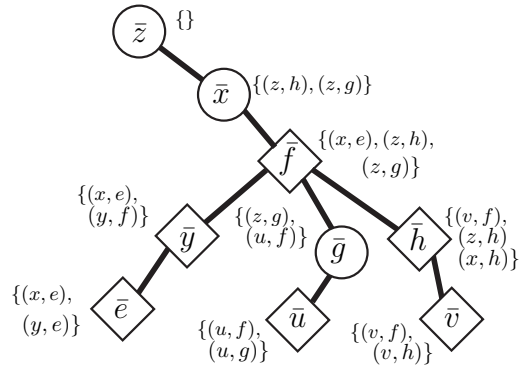


Figure 3: The cluster tree corresponding to Figure 2, showing the boundary of each cluster.

3.2 Computing Marginal Distributions

As with bucket elimination, the root of the cluster tree provides the marginal function for whatever variable is removed last. Moreover, it is also straightforward to compute the marginal at any other vertex by propagating information downward through the cluster tree. We compute the marginal distribution of a node v as follows.

Let $\partial_T\bar{u}$ be the set of tree edges on the boundary of $\bar{u}$, i.e. $\partial_T\bar{u} = \partial\bar{u} \cap E_T$, and let $v_1, \ldots, v_n$ be the sequence from $\bar{v}$ to the root ($v_1 = \bar{v}$, v_n the root). We compute a downward pass of marginalization functions from v_n to v_2 as

$$M_{\bar{v}_i}(\cdot) = \sum_{X \setminus X_{\bar{v}_{i-1}}} \psi_{v_i}(\cdot) \prod_{\bar{u} \in A_{\bar{v}_i}} \varphi_{\bar{u}}(\cdot) \prod_{\bar{a} \in B_{\bar{v}_i}} M_{\bar{a}}(\cdot)$$

where $A_{\bar{v}_i} = S_{\bar{v}_i \setminus \{\bar{v}_{i-1}\}}$ is the set of children of $\bar{v}_i$ which are not on the path from $\bar{v}$ to the root, and $B_{\bar{v}_i}$ defined in terms of the tree edges as follows. If $\partial_T\bar{v}_i \setminus \partial_T\bar{v}_{i-1} = \{(a_1, a_1'), \ldots, (a_t, a_t')\}$ with $a_1', \ldots, a_t' \in \bar{v}_i$, then $B_{\bar{v}_i} = \{\bar{a}_1, \ldots, \bar{a}_t\}$. We know by the properties of the hierarchical clustering that each $\bar{a}_i \in B_{\bar{v}_i}$ is an ancestor of $\bar{v}_i$ in the cluster tree.

Each of these "messages" from parent $\bar{v}_i$ to child $\bar{v}_{i-1}$ is computed using only information on (messages into) the path above $\bar{v}_i$. The marginal at node v is computed as

$$g^v(X_v) = \sum_{X \setminus X_v} \psi_v(\cdot) \prod_{\bar{u} \in S_{\bar{v}}} \varphi_{\bar{u}}(\cdot) \prod_{\bar{a} \in B_{\bar{v}}} M_{\bar{a}}(\cdot),$$

combining the information above and below $\bar{v}$.

In the previous work [2], the combination of G being tree-structured and the selection criteria for creating clusters via rake or compress operations ensured that the computational complexity of each of these calculations was limited. For graphs with cycles, we shall see that these computations may grow more complex (due to the additional "non-tree" edges), but are still bounded and can be controlled sufficiently well to yield practically useful algorithms.

4 A Cluster Tree Data Structure

In this section, we describe a data structure for computing marginal distributions and performing various changes to the structure of the graphical model efficiently.

The idea behind our data structure is to maintain a balanced clustering of a factor graph. To do this, we require the user provide a factor graph along with a spanning tree (or forest) for that graph. We then build a hierarchical clustering of the factor graph, in which the specified spanning tree defines the tree edges E_T of the clustering. Using this representation, we can perform marginal queries in time proportional to the depth of the cluster tree and to the size of the cluster functions stored at each node.

To compute and maintain a balanced clustering, we use the RC-Tree (Rake-and-Compress) tree data structure [1, 3]. This data structure constructs a hierarchical clustering of a tree by performing rake and compress operations and guarantees that the clustering has an expected depth of $O(\log n)$ in the size of the tree. The RC-Tree itself mimics the structure of the clustering: each node is a cluster and there is an edge from a cluster/node to its immediate subclusters. Thus, it enables traversing the clustering like an ordinary tree. In addition to these operations, RC-Trees enable inserting and deleting tree edges and updating the hierarchical clustering so that it remains balanced under any change to the underlying tree.

Since we work with general factor graphs, however, the RC-Tree representation itself does not suffice (RC-Trees are sufficient only for tree-structured factor graphs). To extend the representation, we follow the techniques described in Sec. 3 for computing the boundaries and cluster functions. More specifically, after building the clustering and its RC-Tree, we annotate each cluster with its set of boundary edges, including both tree and non-tree edges, and compute its cluster function as a partial marginalization of its factors over all variables except those on the boundary.

With an RC-tree annotated with boundaries and cluster functions, we can query the data structure to compute marginal functions in the manner described in Sec. 3.2.

To support changes to the underlying structure efficiently, we explicitly distinguish between tree edges and non-tree edges and we require that the spanning tree is kept consistent under changes. This requires, for example, that the user does not delete a spanning tree edge unless the graph becomes disconnected (i.e., there cannot be non-tree edges crossing the cut defined by that tree edge). In other words, the user is responsible for ensuring that the connectivity of the tree-edges matches the connectivity of the factor graph as a whole. This approach makes our interface somewhat crowded, but there is a reason: we wish to provide complete control to the user about the particular spanning tree being maintained, since this is crucial to performance (as we describe in Sec. 4.1). We note that distinguishing between tree and non-tree edges places no restrictions as to what changes can be performed, and the user can still insert and delete any edge. We simply require that if a tree edge is to be removed, it be replaced by another tree edge (perhaps by promoting a non-tree edge) unless its two endpoints are not connected via any other path. We handle changes to the structure of the factor graph as follows.

Replacing a factor: To replace a factor f, we first change it in the input factor graph. We then find the cluster $\bar{f}$ that identifies f in the RC-Tree and update all cluster functions on the path from $\bar{f}$ to the root. Since each cluster function depends only on its subclusters, this sequence of updates suffices.

Insert/delete non-tree edges: Let (u, f) be the non-tree edge being inserted or deleted. We first insert/delete (u, f) into/from the input factor graph. We then find the clusters $\bar{u}$ and $\bar{f}$ in the RC-Tree and visit their ancestors in a bottom-up traversal. When visiting a cluster, we update its boundary edges, which may now need to be changed to exclude (u, f) and recompute its cluster function based on its changed boundary. Since only ancestors of $\bar{u}$ and $\bar{v}$ may have (u, v) as a boundary edge, updating only the ancestors suffices.

Insert/Delete tree edges: Let (u, f) be the tree edge being inserted or deleted. We first insert/delete (u, f) into/from the factor graph as requested. We then insert/delete (u, f) from the spanning tree and use the change-propagation method supplied by the RC-Tree to update the clustering [1]. Change-propagation will update the RC-Tree by deleting some of the existing clusters and inserting some new clusters. We compute the boundaries and the cluster functions for newly created clusters by starting at the root(s) of the RC-Tree(s) involved in the operation and performing a top-down traversal until we visit all new clusters. It is a property of the RC-Tree data structure that all new clusters can be found in this way.

We note that it is for simplicity of presentation that we assume operations consisting of only single changes—multiple changes can be performed simultaneously.

4.1 Interface and Efficiency

We briefly describe the concrete interface to our data structure and analyze the running time for these operations.

The interface supports the following operations: ***cluster(G,T)***, ***query(v)***, ***replaceFactor(old, new)***, ***insertTreeEdge(e)***, ***deleteTreeEdge(e)***, ***insertTreeEdge(e)***, ***deleteNonTreeEdge(e)***. The ***cluster*** operation takes a factor graph G and a spanning tree T of G and constructs hierarchical clustering. The ***query*** operation takes a vertex

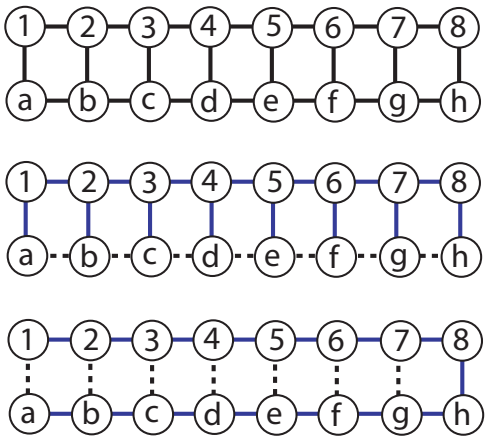


Figure 4: A pairwise factor graph (only variables shown) and two possible spanning trees (shown with thick edges). The first tree results in low measure $\mu_{T_1}(G) = 3$, but the second does not ($\mu_{T_2}(G) = 8$).

of the factor graph and returns the marginal of the vertex. The ***replaceFactor*** operation replaces a factor with another factor. The rest of the operations insert or delete edges in the input factor graph.

To analyze the efficiency of our data structure, we define a notion of the *measure* of a factor graph and its spanning tree. Let G be a factor graph and T a spanning tree; we first define the measure of an edge $e \in T$, written $\mu_T(e)$, as one plus the maximum size of the number of non-tree edges that cross a cut defined by e. More precisely, for an edge e from T, let T_e and T'_e be the components of T separated by deletion of e. Let G_e and G'_e be the subgraphs of G induced by the vertices of T_e and T'_e respectively. Then $\mu_T(e)$ is the size of the cut between G_e and G'_e. The measure of G with respect to T, written $\mu_T(G)$, is the maximum-sized cut over all edges in T.

The importance of this measure is that it helps bound the size of the boundary for a cluster: if the number of tree edges that belong to the boundary of a cluster is b, then the boundary size is at most $b \cdot \mu_T(G)$. Since we use tree contraction to construct the cluster tree, our clusters have at most two tree edges in their boundary. Thus, the boundary of any of our clusters is at most $2\mu_T(G)$.

Fig. 4 shows a pairwise graphical model (top), with factors omitted (one for each edge), and two different spanning trees for it (middle and bottom) with spanning tree edges are highlighted. The factor graph has measure 3 with respect to the first spanning tree because removing any tree edge results in a cut of size at most 3. For example, for the edge $(4, d)$ the cut size is 3—it separates d from the graph, which has two incident non-tree edges. Other vertical tree edges behave equivalently, and for the horizontal tree edges, the cut size is two. Thus for the first spanning tree the measure of the graph is small. For the second spanning tree, however, the measure is large. In particular, removing the edge $(8, h)$ separates the graph into two components consisting of the vertices at the top and those at the bottom with 8 cross edges. This example can be generalized to n nodes such that the measure with respect to this kind of a spanning tree is $n/2$.

By allowing the user to choose the particular spanning tree being used, our data structure allows the measure of the graph to be kept small. This is important because as we prove in the next section, the measure the complexity of our data structure depends exponentially on β. In essence, these differences correspond to a good or poor choice of triangulation in the junction tree algorithm, or elimination orderings in bucket elimination. For these algorithms, good heuristics have been found by researchers over time, and are generally applied in an application-dependent manner.

For a factor graph G and a spanning tree T, let d be the domain of its variables and let k the maximum degree of its nodes. We define the constant *characteristic* of G, denoted α, as the constant $\alpha = d^{k+1}$. Note that representing an (input) factor itself may require this much space.

For the analysis consider some graphical model G with spanning tree T, measure $\beta = \mu_T(G)$ and characteristic α. Our bounds are in terms of the the characteristic and measure of G. For the bounds we assume that degree of the input graph k and domain size of the variables d are positive constants.

Our key lemma, stated below, bounds the time for computing the boundary and cluster function of a cluster.

Lemma 4.1 (Cluster Cost) *The boundary and cluster function of any cluster can be computed in $O(\alpha^\beta)$ time.*

Proof: We first note that since each cluster has at most two tree edges, it has a boundary of at most 2β edges.

Consider computing the boundary for some cluster. We will first bound the number of edges participating in the boundary computation. These edges consists of the boundary edges of the subclusters, the edges between the subclusters and the identifier vertex, and the boundary edges of the cluster itself. For counting purposes, suppose we place a pebble at each end point. The number of pebbles contributed by the k subclusters is $2k\beta$. The number of pebbles contributed by the edges between the identifier and the subclusters is k, because the other endpoints of these edges are inside the clusters and already counted. Finally the pebbles contributed by the boundary edges of the cluster itself is 2β because one end point of the boundary edges is inside subclusters. The total number of edges is half the size of the pebbles, i.e., $\frac{2k\beta+2\beta+k}{2} = (k+1)\beta + \frac{k}{2}$. By maintaining sorted boundaries and performing a $(k+1)$-way merge technique, we can compute the boundary for the cluster in $O\left(((k+1)\beta + \frac{k}{2}) \log k\right)$ time. This running time is negligible compared to that of computing the cluster

function, described next.

For computing the cluster function note that there can be at most $(k+1)\beta + \frac{k}{2}$ boundary variables, because each edge is incident on one variable. The combined domain of these variables then has size at most $d^{(k+1)\beta+\frac{k}{2}}$. We can compute the cluster functions by considering each member of the combined domain and performing k additions or multiplications, giving total time $O(k \cdot d^{(k+1)\beta+\frac{k}{2}})$. ■

Theorem 1 (Hierarchical Clustering) *Consider a factor graph G with n nodes and with spanning tree T. Let α be the characteristic of G and let β be the measure of G with respect to T. We can compute the cluster tree of G in $O(\alpha^\beta \cdot n)$ time The resulting cluster tree has n clusters and expected $O(\log n)$ depth where the expectation is taken over internal randomization.*

Proof: It is known that the cluster tree can be computed in expected $O(n)$ time, independent of the cluster functions and boundaries [1, 3], and that the depth of the cluster tree is $O(\log n)$ in expectation. Since computing the boundary and the cluster function for each cluster takes $O(\alpha^\beta)$ time, the bound follows. ■

We now state the theorem for queries and dynamic changes. Due to space restrictions, we omit the proofs here. Both theorems follow from the fact that changes and queries require traversing a path from the root to an update or a query node while perhaps updating cluster functions and boundaries or computing marginalization functions, which can be performed in $O(\alpha^\beta)$ time.

Theorem 2 (Marginal Queries and Dynamic Changes) *Consider a factor graph G with n nodes and with spanning tree T. Let α be the characteristic of G and let β be the measure of G with respect to T. We can compute the the marginal of a variable in $O(\alpha^\beta \log n)$ expected time. Similarly each dynamic change can be processed in expected $O(\alpha^\beta \log n)$ time.*

5 Experimental Results

We compare the performance of a Matlab implementation of our algorithm to a standard implementation of a junction tree-based sum-product algorithm provided by the Bayes' Net Toolbox (BNT) [12]. We examine the speed-up provide by adaptive inference in two scenarios: synthetic data, which provides some control over the graph size and tree-width of the problems, and graphical models constructed from known protein backbone structures.

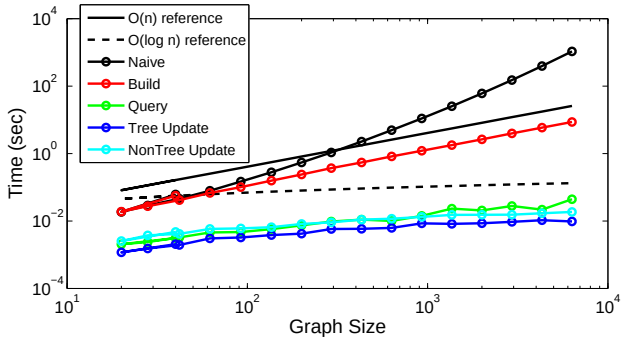


Figure 5: **Synthetic data.** Log-plot of the runtime of naive sum–product on a junction tree (BNT) versus our algorithm. Average update times (over 500 trials) are two to four orders of magnitude faster than performing inference from scratch.

5.1 Synthetic Data

For our synthetic data set, we randomly generated factor graphs with n variables and m factors, where $50 \leq n \leq 1000$, and $m = n - 1$. We initialize each input graph to be a simple Markov chain, where each factor f_i depends on variables x_i and x_{i+1}, where $1 \leq i < n$. This chain comprises the set of tree edges in our algorithm. Then, for given parameters k and ℓ, we add cycles by adding non-tree edges as follows: if i is a multiple of k, we add variable x_i to factor $f_{i+\ell-1}$ to create a cycle of length ℓ. This creates a fairly structured yet loopy graph with limited tree-width.

The results for these synthetic experiments are shown in Fig. 5. We initially compute the (wall clock) time required to construct the cluster tree of the graph (using $k = 2$ and $\ell = 2$). To preserve the predictable tree-width of the problem, updates to the graph structure are performed in pairs by selecting a non-tree edge at random, removing it, updating the cluster tree, adding the edge back in and updating the cluster tree again. We also measure the time to query the marginal at a particular variable as well as the time to update factor definitions (i.e., the values of the factor and not the number of variables it depends on).

We find that our build time is slightly faster than direct inference using the BNT, possibly due to differences in elimination ordering, implicit (cluster tree) vs. explicit (junction tree) maintenance of the tree-decomposition, or simply differences in Matlab programming choices. Most importantly, we see that all of our update operations exhibit average running times (over 500 trials) that are logarithmic in n, and are between one to three orders of magnitude faster than performing inference from scratch.

5.2 Application to Protein Structure

Graphical models constructed from protein structures have been used to successfully predict structural properties [15]

Protein	Size	BNT	Build	Query	Update	**Speedup**
1aie	31	0.213	0.165	0.008	0.012	**8.24**
1nkd	59	0.422	0.252	0.011	0.012	**18.0**
1orc	64	0.504	0.486	0.084	0.064	**3.39**
1vqb	86	0.782	0.469	0.072	0.047	**6.57**
1rzl	91	0.885	0.505	0.068	0.061	**6.86**

Figure 6: **Five proteins from the SCWRL benchmark.** Running times for an implementation of junction tree in BNT, and times for building, updating, and querying using our algorithm. Updates to factors and addition/removal of edges can be applied 3–18 times faster than recomputing from scratch.

as well as free energy [9]. These models are typically constructed by taking each node as an amino acid whose states represent *rotamers* [7], and basing conditional probabilities on a physical energy function (e.g., [14, 4]). A typical goal of using these models is to efficiently compute a maximum–likelihood (i.e. low–energy) conformation of the protein in its native environment. Updating factors allows us to study, for example, the effects of amino acid mutations, and the addition and removal of edges corresponds directly to allowing backbone motion in the protein. Furthermore, the effect of these updates on the model can then be incorporated in logarithmic time, which was not possible in previous approaches.

To test the feasibility of our algorithm for these applications, we constructed factor graphs from five moderately-sized proteins drawn from the SCWRL benchmark [4]. For each protein, we constructed the a factor graph by taking each amino acid as a variable, adding interactions between sequential amino acids as tree edges and steric interactions as non-tree edges. We performed the same updates as for our synthetic test set above. The table in Fig. 6 shows the results of our experiments. We see that for queries and updates, our approach gives a speedup of 3–18 times over inference from scratch. These results are consistent with our synthetic experiment above (i.e., graphs with just under a hundred variables), and show that an adaptive approach to inference can be useful in modeling protein structure. We note however, that for larger proteins, our choice of spanning tree (simply the protein backbone) produced graphs whose treewidth was too large for either our algorithm or sum-product. We are currently exploring other methods for choosing the spanning tree in a protein factor graph (e.g., based on rigid secondary structure elements).

6 Conclusion

We describe an efficient algorithm for adaptive inference in general graphical models. Our algorithm constructs a balanced representation of a spanning tree of the input graphical model, and represents cycles in the model by annotating this data structure. We can support all updates and marginal computations in expected $O(\alpha^\beta \log n)$ time, where α is a constant and β is the size of a particular graph cut. Our experiments show that approach provides significant speedups on both synthetic and real protein data.

References

[1] U. Acar, G. Blelloch, R. Harper, J. Vittes, and M. Woo. Dynamizing static algorithms with applications to dynamic trees and history independence. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2004.

[2] U. Acar, A. T. Ihler, R. R. Mettu, and Ö Sümer. Adaptive Bayesian inference. In *Proc. NIPS*. MIT Press, 2008.

[3] U. A. Acar, G. Blelloch, and J. Vittes. An experimental analysis of change propagation in dynamic trees. In *Proc. 7th ACM-SIAM W. on Algorithm Eng. and Exp'ts*, 2005.

[4] A. A. Canutescu, A. A. Shelenkov, and R. L. Dunbrack Jr. A graph-theory algorithm for rapid protein side-chain prediction. *Protein Sci*, 12(9):2001–2014, Sep 2003.

[5] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 75–104. MIT Press, 1998.

[6] A. L. Delcher, A. J. Grove, S. Kasif, and J. Pearl. Logarithmic-time updates and queries in probabilistic networks. *J. Artificial Intelligence Research*, 4:37–59, 1995.

[7] R. L. Dunbrack Jr. Rotamer libraries in the 21st century. *Curr Opin Struct Biol*, 12(4):431–440, 2002.

[8] S. E. Fienberg. An iterative procedure for estimation in contingency tables. *Ann. Math. Stat.*, 41(3):907–917, 1970.

[9] H. Kamisetty, E. P Xing, and C. J. Langmead. Free energy estimates of all-atom protein structures using generalized belief propagation. In *Proc. 11th Ann. Int'l Conf. Research in Computational Molecular Biology*, pages 366–380, 2007.

[10] F. Kschischang, B. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory*, 47(2):498–519, February 2001.

[11] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems. *J. Royal Stat. Society, Ser. B*, 50:157–224, 1988.

[12] K. Murphy. The Bayes net toolbox for Matlab. *Computing Science and Statistics*, 3:1–20, 2001.

[13] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman, San Mateo, 1988.

[14] S. J. Weiner, P.A. Kollman, D.A. Case, U.C. Singh, G. Alagona, S. Profeta Jr., and P. Weiner. A new force field for the molecular mechanical simulation of nucleic acids and proteins. *J. Am. Chem. Soc.*, 106:765–784, 1984.

[15] C. Yanover and Y. Weiss. Approximate inference and protein folding. In *Proc. NIPS*, pages 84–86, 2002.

Identifying reasoning patterns in games

Dimitrios Antos
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
dantos@eecs.harvard.edu

Avi Pfeffer
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
avi@eecs.harvard.edu

Abstract

We present an algorithm that identifies the reasoning patterns of agents in a game, by iteratively examining the graph structure of its Multi-Agent Influence Diagram (MAID) representation. If the decision of an agent participates in no reasoning patterns, then we can effectively ignore that decision for the purpose of calculating a Nash equilibrium for the game. In some cases, this can lead to exponential time savings in the process of equilibrium calculation. Moreover, our algorithm can be used to enumerate the reasoning patterns in a game, which can be useful for constructing more effective computerized agents interacting with humans.

1 INTRODUCTION

Games are strategic interactions between agents that have different capabilities, information and objectives. Such games, which often involve uncertainty about the world, have become vitally important in computer science as the basis for describing multiagent interaction. The standard solution concept for games is Nash equilibrium. Solving a game is usually taken to mean finding a Nash equilibrium which will specify the strategies of agents. Unfortunately, however, computing Nash equilibrium is a serious bottleneck to using the game theoretic approach; as Daskalakis et al. [2006] have shown, calculating a single Nash equilibrium is PPAD complete. The UAI community has developed graphical representations such as graphical games [Kearns et al. 2001], action graph games [Jiang and Leyton-Brown 2006] and multi-agent influence diagrams (MAIDs) [Koller and Milch 2001], all of which have structural properties that assist in the solution of games. Even in these frameworks, however, computing Nash equilibria can still be very hard.

This paper addresses this bottleneck in two ways. First, it presents a method for simplifying a game in order to make it easier to solve. The method works by analyzing a MAID to discover the reasoning patterns that apply in the given situation. A reasoning pattern is a form of argument that can lead to or motivate a decision. Pfeffer and Gal [2007] showed that all reasoning patterns in MAIDs fall into one of four graphical categories. Furthermore, they showed that if no reasoning pattern holds for a particular decision, the agent making the decision has no reason to prefer one action over another.

We present an algorithm that identifies whether reasoning patterns hold for different decisions, and simplifies the game if they do not. Our algorithm relies on the definitions of reasoning patterns in [Pfeffer and Gal 2007] and also integrates insights of Koller and Milch [2008] that identify cases in which edges can safely be removed from a MAID. We use an iterative procedure in which the MAID is repeatedly simplified. We prove that our algorithm is correct and that it always results in a maximally simplified MAID. In particular, the order in which nodes and edges are considered for removal does not matter. We show that our algorithm has polynomial running time, and also demonstrate how memoization improves the complexity of the algorithm. We present an example showing that our algorithm leads to significant savings in the cost of solving a MAID, in some cases exponential savings.

The second way we address the bottleneck is by providing support for non-equilibrium decision making. Here the goal is to develop good strategies for games which are not necessarily Nash equilibria. We extend our algorithm to identify all the reasoning patterns in a game. This might be helpful in a number of ways. First, the reasoning patterns could be presented to a human decision maker who makes the ultimate decisions. By examining the reasoning patterns, the decision maker can weigh the different arguments for

making decisions and come up with a good decision, based on his or her beliefs about how others are making their decisions. Second, reasoning patterns might provide the basis for automatic approximate solution of games. Instead of solving a single large game, one might solve a number of simpler games and combine the solutions to produce strategies for the large game. Third, reasoning patterns can help characterize how people actually play in games, and thereby help in developing strategies that best respond to people's play. For example, the reasoning pattern known as signaling brings about issues of trust, while the pattern known as manipulation focuses agents on the reciprocal aspects of their interaction. By making these social aspects of reasoning explicit, reasoning patterns facilitate understanding and modeling behavior. Finally, even in games in which an equilibrium can be computed, reasoning patterns may be useful in explaining the equilibrium to human decision makers. Instead of a dry suggestion "choose decision rule δ because it is a Nash equilibrium," the computer decision support system will be able to say "choose δ because it will encourage another agent to help you without sacrificing too much of your own utility."

2 PRELIMINARIES

Multi-Agent Influence Diagrams (MAIDs) are an extension of Influence Diagrams to the multi-agent case [Koller and Milch 2001]. A MAID is a directed acyclic graph with three types of nodes: *Chance* nodes, represented as circles, contain conditional probability distributions over their domains given their inputs. *Decision* nodes, represented as rectangles, are used to denote choices belonging to particular agents. Edges incoming to a decision node reveal the information that is available to the agent at the time of his decision. Finally, *utility* nodes, which are diamond-shaped and also belong to particular agents, represent deterministic functions from the values of their parents to real numbers. A *strategy* σ_A for an agent A in a MAID is a set of decision rules, one for each decision node belonging to A, which maps each configuration of its parents $\mathbf{Pa}(D)$ to a probability distribution over the actions in its domain $\mathrm{Dom}(D)$. Similarly, a *strategy profile* σ for a MAID is a set containing one strategy for each decision in the MAID. A strategy profile σ defines a probability distribution over all the MAID's nodes $P^\sigma(\mathbf{C}, \mathbf{D}, \mathbf{U}) = \prod_{c_i} Pr(c_i|\mathbf{Pa}(c_i)) \prod_{d_j} \sigma_j(d_j|\mathbf{Pa}(d_j)) \prod_{u_k} \mathrm{I}[u_k = U_k(\mathbf{Pa}(u_k))]$, where $\mathbf{C}$, $\mathbf{D}$ and $\mathbf{U}$ are the chance, decision and utility nodes in the MAID and $\mathrm{I}[x]$ is the indicator of x. The *expected utility* of an agent A under strategy profile σ is given by $\mathbf{EU}^\sigma(A) = \sum_{U \in U_a} \sum_u Pr^\sigma(U = u)u$. Solving a MAID requires calculating a Nash equilibrium, i.e., a strategy profile such that no agent may hope to increase his expected utility by unilaterally deviating from his chosen strategy.

In their paper, Koller and Milch [2008] describe an algorithm for simplifying the structure of a MAID by removing edges that are incoming to decision nodes. In particular, their method relies on the fact that, whenever a parent of a decision node is d-separated from its utilities given the decision and its other parents, then one can remove the edge connecting this parent to the decision. Our approach for simplifying a game is based on examining the *reasoning patterns* of agents. These are surprisingly simple graphical properties that all decision nodes are expected to possess in a MAID, if the strategies of these nodes are important to its solution. We now define the concepts of a motivated and effective decision node:

Definition 1. *A decision node D of agent A is called* motivated *if, for some configuration q of its parents, $\mathbf{EU}^{<\sigma_{-A}, d_1>}(A|q) \neq \mathbf{EU}^{<\sigma_{-A}, d_2>}(A|q)$ for $d_1, d_2 \in \mathrm{Dom}(D)$.*

Intuitively, a motivated node is one where the agent cares about his decision. A non-motivated node, by contrast, is one where the agent has no reason to choose one action over another, no matter what the information available to him might be.

Definition 2. *A decision node d of agent a is called* effective *if it participates in at least one reasoning pattern.*

We briefly and informally present the four reasoning patterns below. The reader is referred to [Pfeffer and Gal 2007] for formal definitions.

- Direct effect: This reasoning pattern is present when the agent can directly or indirectly (but not through another agent's decision) influence his own utility. An example of direct effect can be seen in Figure 1a.
- Manipulation: Here an agent A may exert influence on another agent B, whose utility A can affect. Agent B is "manipulated" to do what A wants her to, through his effect on her utility. See Figure 1b for an example of manipulation.
- Signaling: Here A has access to information that is valuable to B, i.e., affects her utility, and may choose an action so as to communicate this information to B. The reason why he might do that is because B, upon seeing his signal, will be expected to choose an action that is favorable to him. An example is shown in Figure 1c.
- Revealing-denying: In this reasoning pattern the agent A has the ability to allow or obstruct the

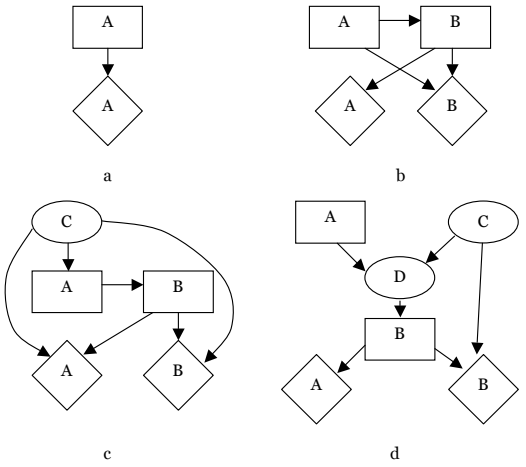


Figure 1: Examples of reasoning patterns

flow of information to another agent B. This way, he effectively increases or reduces the level of certainty B has when making her decision, and thus he may elicit a more favorable response to him. An example is shown in Figure 1d.

To obtain their results, Pfeffer and Gal restrict the strategy space by considering only *well-distinguishing* (WD) *strategies*. This set contains all strategies that do not differentiate between actions based on information that is irrelevant to the decision. Pfeffer and Gal show that WD strategies always contains a Nash equilibrium. Moreover, under WD strategies, the following theorem holds:

Theorem 1. *Assuming all agents play WD strategies, if a decision node is motivated, then it is always effective* (proven in [Pfeffer and Gal 2007]).

3 SIMPLIFYING MAIDs

The discussion of reasoning patterns suggests that, under some circumstances, some decision nodes will be unmotivated. In this section we present an algorithm for identifying unmotivated nodes; this will allow us to simplify a MAID. We will be able to turn these nodes into chance nodes and remove all edges incoming to them.

The algorithm, presented as Algorithm 1, receives as input a MAID M and returns a simplified version of it. It works by iteratively going through two phases: The first (lines 6-17) is a reasoning pattern identification phase, where for each decision node the algorithm tries to establish that it is *motivated*, i.e., at least one reasoning pattern holds for it. Non-motivated decision nodes are those in which any action taken has no effect on the decision maker's payoff. These are replaced by chance nodes with uniform conditional probability distributions and any edges incoming to them are removed. The second phase (lines 18-20) is an edge pruning phase, which uses the algorithm of [Koller and Milch 2008].

Algorithm 1 The simplification algorithm

Input: MAID G

1: $D \leftarrow$ decision nodes in G
2: **for** d in D **do**
3: $\quad effective(d) \leftarrow true$
4: **repeat**
5: $\quad retracted \leftarrow false$; $simplified \leftarrow false$
6: $\quad$ {identification phase}
7: $\quad$ **repeat**
8: $\quad\quad changed \leftarrow false$
9: $\quad\quad$ **for** d in D **do**
10: $\quad\quad\quad$ **if** not (**df**(d) or **man**(d) or **sig**(d) or **rev**(d)) **then**
11: $\quad\quad\quad\quad effective(d) \leftarrow false$
12: $\quad\quad\quad\quad simplified \leftarrow true$
13: $\quad\quad\quad\quad changed \leftarrow true$
14: $\quad\quad\quad\quad$ remove edges incoming to d
15: $\quad\quad\quad\quad$ make d a chance node w/uniform distr.
16: $\quad\quad$ discard memoization database
17: $\quad$ **until** $changed = false$
18: $\quad$ {pruning phase}
19: $\quad$ **if retract_edges**(G) **then**
20: $\quad\quad retracted \leftarrow true$
21: **until** $retracted = simplified = false$

The above algorithm uses procedures **df**, **man**, **sig**, **rev** and **retract_edges**, which are defined below. The first four of these correspond to detecting the existence of the respective four reasoning patterns, whereas the fifth implements the algorithm in [Koller and Milch 2008]. The algorithm terminates when a full iteration (identification and pruning phases) causes the graph to remain unchanged.

All these procedures are built upon simple graph reachability or d-separation operations, all of which can be efficiently computed in polynomial time. For example, the procedure **directedDecisionFreePath**(x, y) is simply implemented by a breadth- or depth-first search. Certain more "sophisticated" procedures are used to search for a path that satisfies particular properties. For example, **effectivePath**(x, y, W) is used to search for a path from x to y on which all decision nodes d have $effective(x) = true$ and, moreover, the path is not blocked by the set of nodes W. Here *blocking* is interpreted as in any Bayesian network.

df(d)

1: $U \leftarrow$ utility nodes belonging to the owner of d
2: **for** u in U **do**

```
    if directedDecisionFreePath(u, D) then
      return true
  return false
```

man(d)

```
U ← utility nodes belonging to the owner of d
N ← decision nodes reachable by d through a directed decision-free path
for u in U and n in N do
  U' ← utilities belonging to the owner of n
  for u' in U' do
    if directedEffectivePath(n, u) and directedEffectivePathNotThrough(d, u', n) then
      return true
return false
```

sig(d)

```
U ← utility nodes belonging to the owner of d
N ← decision nodes reachable by d through a directed decision-free path
for u in U and n in N do
  U' ← utilities belonging to the owner of n
  w' ← all parents of n that are not descendants of d
  for u' in U' do
    if directedEffectivePath(n, u) then
      A ← ancestors of d
      for a in A do
        if backDoorPath(a, u', w') then
          w ← all parents of d that are not descendants of a
          if effectivePath(a, u, w) then
            return true
return false
```

rev(d)

```
U ← utility nodes belonging to the owner d
N ← decision nodes reachable by d through a directed decision-free path
for u in U and n in N do
  U' ← utilities belonging to the owner of n
  w ← all parents of n that are not descendants of d
  for u' in U' do
    if directedEffectivePath(n, u) and frontDoorIndirectPath(d, u', w) then
      return true
return false
```

retract_edges(d)

```
InfEdges ← all edges incoming to decision nodes in G
removed ← false
for (x, y) in InfEdges do
  disabled((x, y)) ← true
D ← all decision nodes in G
repeat
  change ← false
  for d in D do
    Parents(d) ← parents of d
    Utilities(d) ← utilities of d
    for p in Parents(d) and u in Utilities(d) do
      w(p, d) ← all parents of d except for p
      if not dSeparUseEnabled(p, u, w(p, d)) then
        disabled((p, d)) ← false
        change ← true; removed ← true
until change = false
remove all disabled edges from G
return removed
```

directedDecisionFreePath(x_1, x_2)

return true if there is a directed, decision-free path from x_1 to x_2

directedEffectivePath(x_1, x_2)

return true if there is a directed path from x_1 to x_2 in which all decision nodes, except perhaps the first node of the path, are effective

effectivePath(x_1, x_2)

return true if there is an undirected path from x_1 to x_2 in which all decision nodes, except perhaps the first node of the path, are effective

directedEffectivePathNotThrough(x_1, x_2, Y)

return true if there is a directed effective path from x_1 to x_2 that does not go through any of the nodes in Y

backDoorPath(x_1, x_2, W)

return true if there is a back-door path from x_1 to x_2 that is not blocked by W

frontDoorIndirectPath(x_1, x_2, W)

return true if there is a non-directed front-door path with converging arrows at some node from x_1 to x_2 that is not blocked by W

dSepartUseEnabled(x_1, x_2, W)

return true if x_1 is d-separated from x_2 given W, by using only edges e having $disabled(e) = false$

A back door path in the above methods is defined as an undirected effective path where the first edge comes into the first node. A front door indirect path is an undirected effective path where the first edge comes out of the first node and, moreover, the path has converging arrows at some node.

3.1 Proof of correctness

We wish to show that our algorithm performs a legitimate simplification M' of the input MAID M, given

our assumptions of how agents reason about available information. We also care about the effect of the order under which nodes are being eliminated, in order to guarantee that the maximum possible number of non-effective nodes are detected and removed.

Definition 3. *A simplification M' of a MAID M is* legitimate *if all the Nash equilibria of M' are also Nash equilibria of M, in the sense that all nodes in M that have not been eliminated do not have an incentive to deviate from their equilibrium strategy in M' and all nodes that are not effective play in M according to a fully-mixed, uniform strategy.*

We begin by first proving that, if the algorithm marks effective and non-effective nodes correctly, all Nash equilibria of the simplified MAID M' are also Nash equilibria of the original MAID. More precisely, since the original game also contains the decision nodes that were eliminated (and replaced by chance nodes) we need to show that extending the equilibria of M' by adding fully mixed uniform strategies for all non-motivated decisions yields a Nash equilibrium of M.

Theorem 2. *Let D be the decision nodes of the original MAID M and D' be the corresponding decision nodes in the simplified MAID M'. If σ' is a Nash equilibrium of M' then construct σ by adding fully mixed uniform strategies for all decision nodes in $D - D'$. Then σ is a Nash equilibrium of M.*

Proof. We prove this by contradiction. Let there be an agent with a decision node a who wishes to deviate from σ_a; there are two cases: either $a \in D'$ or $a \in D - D'$. In the first case, if the agent owning a wants to deviate to a strategy σ_a^1 in M then she would deviate to σ_a^1 in M' as well, contradicting our assumption that σ' is a Nash equilibrium in M'. In the second case, a was marked as non-effective, therefore a is not motivated, by Theorem 1. By the definition of a non-motivated node, $\mathbf{EU}^{<\sigma_{-a},d_1>}(a,\mathbf{q}) = \mathbf{EU}^{<\sigma_{-a},d_2>}(a,\mathbf{q})$ for every pair of actions d_1, d_2 of a and every configuration $\mathbf{q}$ of its informational parents. Therefore a provides with a fully mixed uniform strategy the same payoff as from any possible deviation from it. Therefore σ is a Nash equilibrium of M. The above reasoning holds even in cases where the agent of a owns other decision nodes besides a, from which he might try to simultaneously deviate. The reason is that a is non-motivated due to strictly graphical properties of the MAID, not due to any particular parameters employed in other chance or decision nodes; thus strategies followed by any other agent—including the owner of a—elsewhere cannot cause it to become motivated. □

We then prove that our algorithm eliminates nodes correctly and maximally, irrespective of order. We do this first by looking at the four procedures that detect reasoning patterns. These work by explicitly following the definitions of the four reasoning patterns, so they are correct (we omit the details). We then look at the first phase of the algorithm (lines 6-18). First, however, we need the following lemma.

Lemma 1. *If R_n^E is the set of reasoning patterns that hold for a decision node n when the set of edges in the MAID is E, then $R_n^{E'} \subseteq R_n^E$ for all $E' \subseteq E$.*

Proof. Consider a MAID with edges E and a set R_n^E of reasoning patterns holding for decision node n. Now remove an edge $e \in E$, such that the MAID now has edges $E' = E - \{e\}$. Suppose now, for the sake of contradiction, that $R_n^{E'} \not\subseteq R_n^E$. This means that a reasoning pattern r did not exist under E but exists under E'. Take the paths P_r of this reasoning pattern and let $E(P_r)$ be the set of their edges. Clearly, $E(P_r) \subseteq E' \subset E$ so all the paths the reasoning pattern r depends on existed in the original MAID with edges E. Thus the only reason why r did not hold under E was that one or more of its paths were blocked at some node. Let $p \in P_r$ be one such path with blocking set W_p ($W_p = \emptyset$ if the definition of the reasoning pattern required no d-separation properties to hold for that path) and let b be the node where p was blocked by W_p. If p has non-converging arrows in b then $b \in W_p$, so p should be blocked again under E', since no nodes were removed, only an edge. If p has converging arrows in b then it means that neither b nor any of its descendants were in W_p. But removing e can neither add b to W_p, nor cause the set of its descendants to grow. Therefore, under E', too, W_p will block p at b and therefore our argument is contradictory. □

Lemma 2. *If a node is identified in some identification phase under some order, it will be so identified under any order.*

Proof. Let $n_1, ..., n_k$ be an order of identifying non-effective nodes and $n'_1, ..., n'_k$ be a different order. Also define as E_n the set of edges in the MAID after node n has been eliminated and as E the edges in the MAID in the beginning, before any elimination takes place. Suppose that under the new order node n_1 is placed at position h. Then, by Lemma 1, in the identification phase and under the new order n_1 will be eliminated irrespective of h, since $E'_h \subseteq E$ and node n_1 was eliminated under E. We then reason by induction. Now assume that $n_1, ..., n_i$ have been eliminated. Then n_{i+1} will be eliminated at the latest in the next phase after all of $n_1, ..., n_i$ have been eliminated, because the set of edges present will be a subset of E_i. □

For the second phase of the algorithm (pruning), this is exactly implemented as in [Koller and Milch 2008], which contains the proof of its correctness.

Theorem 3. *The algorithm produces a correct and maximal simplification of a MAID.*

Proof. We know that the operation of each phase is correct. Moreover, we know the pruning phase only removes edges. Thus, by Lemma 1, it does not matter in the context of the identification phase on which iteration of the algorithm the pruning phase removes an edge e, as long as it eventually removes it (on some iteration). Thus the only thing we need to establish is that no operation in the identification phase might ever prevent an edge from being removed in the pruning phase.

The pruning phase works by testing for certain d-separation properties, while the identification phase only removes edges (never adds). Thus, in spirit similar to the proof of Lemma 1, if the MAID has edges E in the beginning of the pruning phase and an edge $e = (x, y)$ is removed during its execution, then if the MAID had edges $E' \subset E$ then e would still be removed. If e was removed with edges E then x was d-separated from the utility nodes of y given $\{y\} \cup \{d : (d, y) \in E, d \neq x\}$, by definition of [Koller and Milch 2008]'s algorithm. Now under the smaller set of edges E' it is the case that x must, again, be d-separated from y's utility nodes, since the removal of any edge in $E - E'$ cannot have made these two less separated. Therefore, no matter in which order we execute the two phases and no matter what their intermediate results are, the end product is the same.

Furthermore, the iteration of identification and pruning phases will terminate. Neither adds an edge and there are at most E edges to remove, so the process will eventually terminate. □

We have shown that our algorithm's operations are consistent with our assumptions and that it will always return a maximally simplified MAID. In the following section we analyze its complexity.

3.2 Algorithm Complexity

The complexity of the algorithm is easy to estimate. We first begin with the bottom-level procedures which are path operations. Those that do not involve a non-empty blocking set are instances of graph reachability, which can be performed in $O(E+N) = O(E)$ time, assuming $E > N$. If the blocking set is non-empty, however, every time a path is expanded one needs to check that it is not blocked at that node. This can be done using an algorithm such as BayesBall [Shachter 1998], which is $O(E)$.

We also use *memoization* to improve the computation of blocking properties along the paths. In particular, whenever we query whether a path with converging arrows at at a node b is blocked by a set W, we store the result $blocked(b, W)$ in a hash table. Subsequent queries for the same node and blocking set first check the hash table for an already computed result and only execute the full operation (costing $O(E)$) if needed. After each iteration, since the structure of the graph has changed, we drop all memoized entries (line 16).

Theorem 4. *The algorithm simplifies the MAID in time $O(D^2N^2E)$, where D is the number of decision nodes in the graph.*

Proof. We have established that all path operations take polynomial time. In particular, suppose C and U are the number of chance and utility nodes. Then procedure **df** performs $O(U)$ simple path operations, so it costs $O(UE)$ in the worst case. Manipulation (**man**) performs $O(DU^2)$ simple operations, for a total cost of $O(DEU^2)$. Signaling (**sig**) requires certain paths to satisfy blocking properties and performs $O(CDU^2)$ of those, for a total worst-case cost of $O(CDU^2E^2)$. Here the E^2 results from the following: For every combination of nodes related to the signaling patterns we need to find certain paths with graph reachability ($O(E)$); for each such path, at every step we need to check for blocking properties, which adds another $O(E)$, for at total of $O(E^2)$. Finally, revealing-denying (**rev**) performs $O(DU^2)$ blocking-sensitive path operations and thus has worst case cost of $O(DU^2E^2)$.

We see that signaling is the most expensive of these operations. However, with memoization, its worst-case complexity can be reduced. We reason as follows. There can be a total of $O(DN)$ blocking sets required for the purposes of identifying reasoning patterns in the graph (for every decision node there can be one blocking set including all of its parents but one, and there are $O(N)$ parents per decision). Thus, even if we were to calculate blocking properties for all nodes and possible sets, the time per iteration would be bounded by $O(DN^2E)$. In a similar fashion, the time complexity for revealing-denying identification can be bounded.

The algorithm as a whole also performs at most $O(D)$ iterations in the outer loop. This is because if two consecutive iterations eliminate no nodes the algorithm by definition terminates, since the pruning phase of the second iteration will remove no edges. Therefore the total cost of the algorithm is polynomial and on the order of $O(D^2N^2E)$. □

Of course it has to be noted that the expected performance of our algorithm is likely to be much better than its worst-case bound calculated above. In particular, the evaluation of the if structure in line 10 is short-circuited, meaning the expensive **sig** operation is only evaluated if **df** and **man** are false, since it is sufficient to show that a node participates in one reasoning pattern to be effective. Moreover, real games very likely have much fewer reasoning patterns mainly consisting

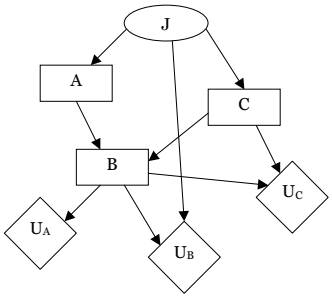


Figure 2: MAID for our simple example

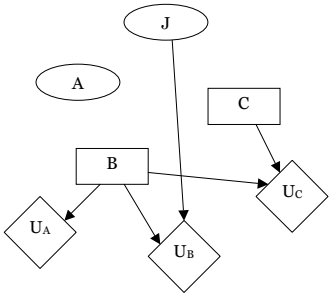


Figure 3: Simplified example MAID

of direct effects and manipulations, especially after the pruning phases of the first iterations have reduced the number of edges in the graph that are important to signaling and revealing-denying patterns.

4 AN EXAMPLE

Consider the following game. Player A is handed a card, which can be either High (H), Medium (M) or Low (L). Another player (B) must guess what type of card A was given; upon a correct guess she wins \$10, otherwise she gets nothing. A can tell her what the value of the card is, not necessarily truthfully. A's payoff is \$10, \$5 or \$2 if B guesses H, M or L, respectively, no matter what the true value of the card is. Finally, another player C has access to the value of the card and must guess what B will guess. If his guess matches B's, he wins \$10, otherwise he gets nothing. Furthermore, B can see C's choice before making her decision. The MAID for this game can be seen in Figure 2.

This game can be solved using a MAID solution algorithm. The algorithm of [Koller and Milch 2001] uses a *relevance graph*. Each strongly connected component of the relevance graph is converted into an extensive form game tree. Unfortunately, in our game the relevance graph contains a single connected component, so the entire game must be converted into a tree. This tree must split at A's, B's and C's choices, resulting in $3^3 = 27$ leaf nodes.

Can we do better? In our card game one might reason as follows: Agent A's utility is not affected by the value of the card J, thus—according to WD strategies—A will not condition his decision on the value of J that he observes. In other words, A is unmotivated and has no reason to tell B the truth about J. Knowing that, B will not believe A. Similarly B will ignore what C tells him, because—again—C's utility is unaffected by the value of J and thus he will ignore it in his decision. The new, simplified MAID can be seen in Figure 3.

Our algorithm in this instance would proceed as follows: During the first iteration, in the identification phase A would be replaced by a chance node (none of the reasoning patterns hold for it) and the edge (J, A) would be removed. In the pruning phase the edge (J, C) is removed, because it is d-separated from U_C given C. Furthermore, since A is d-separated from U_B given $\{B, C\}$ the edge (A, B) would be removed as well. For the same reason (C, B) would be pruned. During the second iteration no change is performed and the algorithm terminates. We now have to solve just two game trees, one for B and one for C, with 9 leaf nodes each.

Now extend this simple game by adding more C-type players. Each of these would have access to J, win upon matching B's choice and have his decision known to B. With n such C-type players, the original game has a tree representation of $3^{(2+n)}$ leaves, whereas running the algorithm for one iteration would simplify this to $n + 1$ trees of 3^2 leaves each; savings in computing the equilibrium of the game are thus exponential.

5 ENUMERATING REASONING PATTERNS

We can easily tweak the algorithm to have it enumerate reasoning patterns as well. This can be done by *not* short-circuiting the **if** of line 10 of the algorithm and changing the functions **df**, **man**, **sig** and **rev** so as not to return when a reasoning pattern is found, but to add it to a list and iterate until no more reasoning patterns of that type are found.

As an example, consider the following two-stage principal-agent game with reputation. In each stage of the game the principal P wants the agent D to execute a task. The agent may be of type *good* or *bad*, but this type is not visible to the principal. The agent's decision is to exert *high* or *low* effort. Bad-type agents enjoy a greater utility from exerting low effort and vice versa for good-type agents. Moreover, the price offered by the principal also affects the agent's decision. Upon the completion (adequate or poor) of

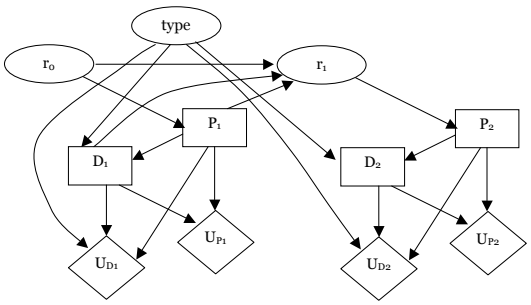


Figure 4: A two-stage principal-agent game

the task in the first stage, the principal moves on to offer another price to the agent for the second task. However, the agent's choice (visible performance) in the first stage is used to update a reputation value (r_0 to r_1), which the principal takes into consideration in his second offer. The reputation at each stage can be *high*, *medium* or *low* and is more likely to change between rounds when it is *medium*. The MAID for the game can be seen in Figure 4 (perfect recall edges are omitted for clarity). Running our algorithm reveals a set of interesting reasoning patterns (the "story" behind each reasoning pattern is provided by us and is *not* automatically generated by the workings of the algorithm):

- Direct effect: All nodes P_1, P_2, D_1, D_2 have direct effect. This corresponds to the fact that the players' actions have an immediate effect on their payoff.
- Manipulation (principal → agent): At each stage the principal manipulates the agent through the price offered. A higher price makes effort = *high* more preferable for the agent and thus indirectly increases the principal's utility.
- Manipulation (agent → principal): D_1 manipulates P_2. This reflects the agent's thinking at stage 1: pretending to be *good* when in fact he is not can cause P_2 to offer him a high price for the second-round task (because the principal trusts him). In other words, this pattern corresponds to deceiving the principal and "milking" your reputation in the second round.
- Signaling (agent → principal): D_1 signals his *type* to P_2 by choosing an appropriate effort level. The variable *type* is not observed by the principal; however, the principal does care about it, since it provides him with insight to the agent's reasoning. The agent, on the other hand, wants his action to reveal something about this hidden piece of information. For example, if the agent is *good*, he wants the principal to know this so that he is offered a high price in the following round.
- Signaling (r_0 to D_1): P_1 signals the current reputation value to D_1. This is subtle but interesting: Suppose the agent believes that the principal has a current reputation value for him (the agent) that is *low*. In that case the agent might consider it a "lost cause" to exert *high* effort, since the principal on the next round is still going to believe he is *bad* and offer him a low price. Conversely, if the agent thinks his current r_0 is *high*, he has less reason to actually try and retain this reputation by exerting *high* effort. Thus, if r_0 is *medium*, the principal wants the agent to know it.
- Revealing-denying (*type* to P_2): P_1, by making a suitable offer to D_1 in the first round, can elicit information about his *type* that will be useful to P_2. For example, if the offer in the first round is too high, then it might be optimal for the agent to always exert *high* effort, which reveals little about his *type*. Experimenting with moderate offers allows the principal to risk guaranteed good performance in the first round, so that he may discover more about the agent's true *type* and exploit that in the second round.

We see in this example how the reasoning patterns can be used to discover the possible motivations or patterns of thought agents may have when considering their decisions. Among them, there are some motivational patterns that are not obvious or easy to detect by means of any reasoning tool available today.

6 DISCUSSION

The algorithm presented in this paper uses the assumption of well-distinguishing (WD) strategies to simplify a game for the purpose of calculating a Nash equilibrium. Despite this being a reasonable assumption for computational agents, it is not clear whether people in general condition their actions only on observations that have an effect on their utilities, or whether they heed otherwise irrelevant signals. We plan to investigate this experimentally.

Another limitation of our algorithm is that it uses strictly graphical properties to discover and remove non-motivated nodes. Although this greatly simplifies the process and keeps it computationally efficient, a more rigorous examination of the parameters inside chance and utility nodes could lead to further simplifications. For example, it might be the case that a *signaling* reasoning pattern does hold graphically, but the parameters of the MAID are such that the first agent (Alice) always optimally decides on one particular action for all possible values of the signal, in which case the second agent (Bob) has no reason to infer anything about it by observing her action and he should rationally ignore it. Another example is zero-

sum games, where selfish rational agents will optimally have no reason to signal any variable of interest to another agent, or believe any such signal.

Moreover, it has to be noted that our algorithm does not quantify the reasoning patterns it discovers. We cannot say, "According to this reasoning pattern, taking action X will increase your utility by such and such an amount." It is up to the decision maker to determine the semantic interpretation and relative importance of the reasoning patterns.

Finally, although in the worst case exponential savings are possible, it is not yet clear how the algorithm performs in a *typical case*. We chose to avoid directly answering this question, mainly because of the challenge in defining "typical cases." Moreover, one should keep in mind that the algorithm's usefulness increases with the size of the game, since in large MAIDs deciding which observation is truly relevant to a decision is not obvious. In such cases, because the algorithm is polynomial—whereas Nash equilibrium computation is not, assuming PPAD $\neq$ P—, it can be run at negligible cost, even if the savings obtained are not significant.

7 CONCLUSION & FUTURE WORK

We have presented an algorithm for identifying reasoning patterns in games. This algorithm can be used in two ways: First, it can identify unmotivated decisions in a MAID graph; these can be effectively ignored for the purposes of calculating a Nash equilibrium, which can sometimes lead to considerable computational savings. Second, it is capable of discovering non-obvious patterns of thought agents might use when making their decisions. These patterns can be presented to a human decision maker to help him or her make good decisions. They may also used as inputs to other algorithms for decomposing, analyzing, explaining or even predicting agent or human behavior.

We believe that in complex systems, and especially those in which human and computer agents interact heavily, it is important for successful agents to model the "context" of their interaction. Certain features of games are likely to bring about different motivations in human behavior, such as reciprocity or cooperation. Reasoning patterns are promising as modeling aids for three reasons: First, they are adequately rich to analyze arbitrarily complex games, yet they are concise in their number. Second, they have a behavioral flavor, in that they talk about *reasoning* and not just utility maximization, yet they also have a rigorous mathematical and normative foundation. And third, they are discoverable in efficient (polynomial) ways. We plan to extend this line of research with the ultimate goal of constructing computerized agents who have a better understanding of human motivations and are better aligned with their goals. This will entail calculating semantic characterizations of the various reasoning patterns, looking at their graphical properties as well as the parameters within the nodes, and explicitly calculating the implications of each pattern to the agents' optimal or equilibrium strategies.

Acknowledgments

The research reported in this paper was supported in part by AFOSR MURI grant FA9550-05-1-0321.

References

[Blum et al. 2006] Ben Blum, Christian R. Shelton, and Daphne Koller, *A Continuation Method for Nash Equilibria in Structured Games*, Journal of Artificial Intelligence Research. 25, pp. 457-502, 2006.

[Daskalakis et al. 2006] Constantinos Daskalakis, Paul W. Goldberg and Christos H. Papadimitriou, *The Complexity of Computing a Nash Equilibrium*, In the 38th ACM Symposium on Theory of Computing, STOC, 2006.

[Jiang and Leyton-Brown 2006] Albert Xin Jiang and Kevin Leyton-Brown, *A Polynomial-Time Algorithm for Action-Graph Games*, In AAAI, 2006.

[Kearns et al. 2001] Michael Kearns, Michael L. Littman and Satinder Singh, *Graphical Models for Game Theory*, In UAI, 2001.

[Koller and Milch 2001] Daphne Koller and Brian Milch, *Multi-Agent Influence Diagrams for Representing and Solving Games*, IJCAI, 2001.

[Koller and Milch 2008] Brian Milch and Daphne Koller, *Ignorable Information in Multi-Agent Scenarios*, Technical report MIT-CSAIL-TR-2008-029, Massachusetts Institute of Technology, Cambridge, MA, 2008.

[La Mura 2000] Pierfrancesco La Mura, *Game Networks*, In UAI, 2000.

[Pfeffer and Gal 2007] Avi Pfeffer and Ya'akov Gal, *On the Reasoning Patterns of Agents in Games*, In AAAI, 2007.

[Shachter 1998] Ross D. Shachter, *Bayes-Ball: The Rational Pastime*, In UAI, 1998.

[Vickrey and Koller 2002] D. Vickrey and D. Koller, *Multi-Agent Algorithms for Solving Graphical Games*, In AAAI, 2002.

Learning Inclusion-Optimal Chordal Graphs

Vincent Auvray
GIGA-R and EE & CS Dept.
University of Liège
Vincent.Auvray@ulg.ac.be

Louis Wehenkel
GIGA-R and EE & CS Dept.
University of Liège
L.Wehenkel@ulg.ac.be

Abstract

Chordal graphs can be used to encode dependency models that are representable by both directed acyclic and undirected graphs. This paper discusses a very simple and efficient algorithm to learn the chordal structure of a probabilistic model from data. The algorithm is a greedy hill-climbing search algorithm that uses the inclusion boundary neighborhood over chordal graphs. In the limit of a large sample size and under appropriate hypotheses on the scoring criterion, we prove that the algorithm will find a structure that is inclusion-optimal when the dependency model of the data-generating distribution can be represented exactly by an undirected graph. The algorithm is evaluated on simulated datasets.

1 INTRODUCTION

A graphical probabilistic model makes use of a graph over random variables to encode a dependency model, i.e. a set of marginal and conditional independence relations. Directed acyclic graphs (DAGs) and undirected graphs (UGs) are two popular classes of graphs used to encode dependency models, leading to graphical models known as Bayesian networks and Markov networks (see [9]).

In this paper, we consider the class of graphical models whose structure is a chordal graph, known as the class of decomposable models. A chordal (or triangulated) graph is an undirected graph where every cycle comprising more than three lines has a chord. The class of dependency models defined by chordal graphs is the intersection of the class of DAG dependency models and the class of UG dependency models. The characterization of the independencies of decomposable models has been exploited in [6] in order to construct algorithms for recovering from independence tests the exact chordal structure of a decomposable model, and to build minimal chordal approximations of UG-isomorphic dependency models.

Despite the chordality restriction on the structure, the class of decomposable models is still fairly large and includes, for example, graphical models with undirected tree structure. Also, exact marginalization using the junction-tree algorithm for probabilistic inference over DAGs and UGs is based on the prior transformation of these graphs into a chordal graph [5].

A greedy hill-climbing search algorithm is often used to learn the DAG structure of a Bayesian Network. Different choices of search spaces and neighborhoods connecting the search space are possible. In particular, the search may proceed over the set of Markov equivalence classes of DAG structures by exploiting the inclusion boundary neighborhood (see [1, 4]). Under appropriate assumptions on the scoring criterion and on the data-generating distribution, a greedy algorithm using this inclusion boundary neighborhood returns an inclusion-optimal structure in the limit of a large sample size (see [2] and [3]). Unfortunately, the size of the inclusion boundary of an equivalence class of a DAG structure is in the worst case exponential in the number of variables, which may prevent the application of this strategy in domains with a large number of variables.

The notion of inclusion boundary neighborhood can also be defined over sets of chordal graphs (see Section 2). In this context, its size is bounded from above by the square of the number of variables (pairs of vertices) and it can be computed easily. In [7], this neighborhood is used to learn the chordal structure of a decomposable Gaussian model with a Monte Carlo procedure.

In this paper, we investigate the optimality properties of the greedy hill-climbing search algorithm using the inclusion boundary neighborhood to learn a chordal structure. We describe a local asymptotic consistency property of scoring criteria that ensures that a greedy search will produce an inclusion-optimal chordal structure when the independence relations holding in the data-generating distribution can be represented exactly by an undirected graph. We conjecture that this property still holds when the independencies of the data-generating distribution can be represented exactly by a directed acyclic graph. Hence, we suggest that inclusion

boundary based learning of chordal models is an interesting avenue for leveraging learning of graphical models to domains with large numbers of variables.

The rest of the paper is organized as follows. Section 2 defines precisely the mechanism by which an undirected graph encodes a dependency model. It also defines and discusses the notions of inclusion-optimality and inclusion boundary. Section 3 introduces a local consistency property for scoring criteria defined over chordal structures and proves that it holds for common criteria such as the BDe score. Our claim that greedy search with the inclusion boundary neighborhood yields inclusion-optimal solutions is proved there. Section 4 presents some experimental results using simulated datasets.

2 BACKGROUND

Consider an undirected graph $G = (X, L)$ whose vertex set X is a set of random variables and whose set of undirected edges (i.e. lines) is denoted by L. Given disjoints sets $A, B, C \subseteq X$, we say that A and B are separated by C in G if all paths between a vertex in A and a vertex in B go through at least one vertex in C. The dependency model encoded by G consists of the set of marginal and conditional independence relations $A \perp B|C$ such that A and B are separated by C in G. In the sequel, we sometimes identify an undirected graph and its dependency model; when we want to distinguish them we will denote by $I(G)$ the dependency model encoded by the graph G.

As mentioned in the introduction, a chordal graph is an undirected graph where every cycle more than three lines long has a chord (see Figure 1).

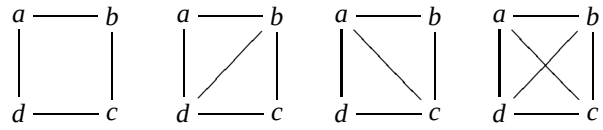


Figure 1: (a) is undirected, but not chordal since it has the chordless cycle a, b, c, d, a of length four. (b), (c) and (d) are all chordal.

Let us define the notion of inclusion-optimality (aka minimal I-mapness in the terminology of [9]) for chordal graphs. Consider a particular dependency model M_0. We say that a *chordal* dependency model M is inclusion-optimal for M_0 if $M \subseteq M_0$ and there is no *chordal* dependency model M' such that $M \subsetneq M' \subseteq M_0$. This notion has a simple graphical interpretation: a chordal graph G encodes an inclusion-optimal dependency model for M_0 if, and only if, (a) it does not encode any independence assumption that does not hold in M_0 and (b) all its proper chordal subgraphs[1] encode such an incorrect independence assumption. For example, suppose that Figure 1(a) encodes M_0. Then, the graphs of Figure 1(b) and Figure 1(c) are both inclusion-optimal chordal graphs with respect to M_0, while the graph of Figure 1(c) is not. As a special case, note that any chordal graph model is the *unique* chordal dependency model which is inclusion-optimal for itself.

To conclude this section, let us present the notion of inclusion boundary in the context of chordal graphs. The inclusion boundary of a chordal graph G is the set of chordal graphs H satisfying

- $I(G) \subsetneq I(H)$ and there is no chordal graph K such that $I(G) \subsetneq I(K) \subsetneq I(H)$, or
- $I(H) \subsetneq I(G)$ and there is no chordal graph K such that $I(H) \subsetneq I(K) \subsetneq I(G)$.

It is straightforward to describe graphically the inclusion boundary of a chordal graph G: it consists of the chordal graphs that differ from G by the addition or removal of a single line. This is a consequence of the fact that, for any two chordal graphs G, H such that H is a subgraph of G, there exists a sequence of chordal graphs $K_0, \ldots, K_n$ such that $K_0 = H$, $K_n = G$ and K_{i+1} is obtained from K_i by adding a single line (see [7]).

3 INCLUSION-OPTIMALITY OF GREEDY SEARCH

In this section, we first introduce a property of *local* consistency for scoring criteria defined over chordal graphs. Then, we show that if a scoring criterion defined over DAG dependency models is decomposable and consistent in the classical sense (see, e.g. [8]) , then it is also locally consistent when restricted to chordal dependency models. Finally, we prove the claim that a greedy hill-climbing search using the inclusion boundary neighborhood and a consistent and locally consistent scoring criterion returns an inclusion-optimal chordal graph when the dependency model of the data-generating distribution can be encoded exactly by an undirected graph.

Following the terminology of [3], we say that a scoring criterion score$(\cdot)$ for chordal graphs is *locally* consistent for a dependency model I if, for any pair of vertices a, b and chordal graphs G, H such that H is obtained from G by removing $a - b$, we have

1. $a \perp b|ne_G(a) \cap ne_G(b) \in I \Rightarrow \text{score}(H) > \text{score}(G)$,
2. $a \perp b|ne_G(a) \cap ne_G(b) \notin I \Rightarrow \text{score}(G) > \text{score}(H)$,

[1] We say that $G' = (X', L')$ is a (proper) subgraph of $G = (X, L)$, iff $X' = X$ and $L \subsetneq L'$, i.e. L is a (proper) subset of L'.

where $ne_K(a)$ denotes the sets of neighboring (i.e. adjacent) vertices of a in K.

Recall that a scoring criterion score(·) for a DAG dependency model encoded by G is decomposable if it can be written as a sum of terms that depend each on only one vertex and its parents, i.e.

$$\text{score}(G) = \sum_{v \in V} f(v, pa_G(v)). \quad (1)$$

The following proposition states that a consistent and locally consistent scoring criterion for chordal dependency models can be obtained from a consistent and decomposable scoring criterion for DAG dependency models. Moreover, it allows us to compute the score difference between neighboring chordal graphs incrementally.

Proposition 1. *If* score(·) *is a scoring criterion over DAG dependency models that is decomposable and consistent for a dependency model I, then it is locally consistent for I when restricted to chordal graphs and*

$$\begin{aligned}\text{score}(G) - \text{score}(H) = {} & f(b, \{a\} \cup (ne_G(a) \cap ne_G(b))) \\ & - f(b, ne_G(a) \cap ne_G(b)), \quad (2)\end{aligned}$$

for chordal graphs G and H such that H is obtained from G by removing the line $a - b$.

Proof. Consider two chordal graphs G and H such that H is obtained from G by removing the line $a - b$. Since H is chordal and does not have $a-b$, the subgraph of H induced[2] by $ne_H(a) \cap ne_H(b) = ne_G(a) \cap ne_G(b)$ is complete. Hence, the subgraph of G induced by $\{a, b\} \cup (ne_G(a) \cap ne_G(b))$ is complete. If $o_1, \ldots, o_k$ is any ordering of $ne_G(a) \cap ne_G(b)$, there exists a perfect ordering o of G starting with $a, o_1, \ldots, o_k, b$. Let K be the DAG obtained from directing the lines of G according to o and let L be the DAG obtained from K by removing $a \to b$. Note that K and L have no v-structure, $\text{score}(G) = \text{score}(K)$, $\text{score}(H) = \text{score}(L)$, $pa_L(b) = ne_G(a) \cap ne_G(b)$ and $pa_K(b) = \{a\} \cup pa_L(b)$. By decomposability of score(·), we thus have

$$\begin{aligned}\text{score}(G) - \text{score}(H) = {} & f(b, \{a\} \cup (ne_G(a) \cap ne_G(b))) \\ & - f(b, ne_G(a) \cap ne_G(b)). \quad (3)\end{aligned}$$

Let A be a complete DAG obtained by orienting the lines of a complete undirected graph according to a vertex ordering starting with $a, o_1, \ldots, o_k, b$ and let B be the DAG obtained from A by removing $a \to b$. We have $pa_B(b) = pa_L(b)$, $pa_A(b) = pa_K(b)$, $\dim(B) < \dim(A)$, $I(A) = \emptyset$ and

$$I(B) = \{a \perp b | ne_G(a) \cap ne_G(b), b \perp a | ne_G(a) \cap ne_G(b)\}. \quad (4)$$

By decomposability of score(·), we have

$$\text{score}(A) - \text{score}(B) = \text{score}(K) - \text{score}(L). \quad (5)$$

[2]The subgraph of $G = (X, L)$ *induced* by $X' \subseteq X$ is the graph $G' = (X', L')$, where $L' = L \cap (X' \times X')$.

By consistency of score(·) for I, the restriction of score(·) over chordal graphs is thus also locally consistent for I. ∎

In practice, scoring criteria over DAG dependency models only satisfy the consistency property asymptotically in the limit of a large sample size. When restricted to chordal dependency models, such scoring criteria will thus only be locally consistent asymptotically.

3.1 Optimality for UG target dependency models

The main result of this paper can now be stated. Its proof relies on results presented in the appendix.

Proposition 2. *If* score(·) *is a scoring criterion for chordal graphs that is consistent and locally consistent for a graph-isomorph dependency model I, then local optima of* score(·) *with respect to the inclusion boundary neighborhood are inclusion-optimal for I.*

Proof. Let G be a local optimum of score(·) with respect to the inclusion boundary neighborhood. Let us show by contradiction that $I(G) \subseteq I$. Suppose that $I(G) \setminus I \neq \emptyset$. Since I satisfies the symmetry, decomposition and intersection properties (see Proposition 3 in the appendix), there exist vertices a and b such that $a \perp b | V \setminus \{a, b\} \in I(G) \setminus I$. Hence, G does not have the line $a - b$. Let us discuss separately the cases where the addition of $a - b$ to G results in a graph H which is chordal and the cases where the resulting graph H is not chordal.

Suppose that H is chordal. Then, H is in the inclusion boundary of G. By strong union, $a \perp b | V \setminus \{a, b\} \notin I$ implies that $a \perp b | ne_H(a) \cap ne_H(b) \notin I$. By local consistency, we thus have $\text{score}(H) > \text{score}(G)$ and G is not a local optimum.

Suppose that H is not chordal. There exists a chordless cycle in H of length ≥ 4. Consider the set of chordless cycles in H of maximum length $m \geq 4$ and the corresponding set of paths in G between a and b of length $n = m - 1 \geq 3$. Let $A_0 = \{a\}$, $A_n = \{b\}$ and, for $i = 1, \ldots, n-1$, let A_i be the set of vertices that can be reached starting from a by hopping along i lines on one of the above paths between a and b. By strong union, $a \perp b | V \setminus \{a, b\} \notin I$ implies that $a \perp b | A_{n-1} \notin I$. By Lemma 4 (see the appendix), there thus exists $i \in \{1, \ldots, n-1\}$ such that $A_{i-1} \perp A_{i+1} | A_i \notin I$ or $a \perp A_i \in I$. Let us discuss the two possibilities separately. First, suppose that $A_{i-1} \perp A_{i+1} | A_i \notin I$. By composition, there exist $u \in A_{i-1}$ and $v \in A_{i+1}$ such that $u \perp v | A_i \notin I$. By chordality of G, note that each set A_i induces a complete subgraph. Hence there is a cycle of length m without chord passing through u and v in H and thus no line between u and v in G. By maximality of this cycle, adding $u - v$ to G results in a chordal graph H' in the inclusion boundary of G. Since $ne_G(u) \cap ne_G(v) \subseteq A_i$, we have $u \perp v | ne_G(u) \cap ne_G(v) \notin I$ by strong union. By local consistency, we thus have $\text{score}(H') > \text{score}(G)$ and G is not

a local optimum. Second, suppose that $a \perp A_i \in I$ for some $i \in \{1, \ldots, n-1\}$ and consider any vertex $u \in A_i$. By decomposition, we have $a \perp u \in I$. There exists a path $p_1, \ldots, p_k$ in G between $p_1 = a$ and $p_k = u$ where no line is a chord. By transitivity, $a \perp u \in I$ implies that $p_j \perp p_{j+1} \in I$ for some $j \in \{1, \ldots, k-1\}$. Since the line $p_j - p_{j+1}$ is not a chord, the graph H' obtained from G by removing $p_j - p_{j+1}$ is chordal and H' is in the inclusion boundary of G. By strong union $p_j \perp p_{j+1} \in I$ implies that $p_j \perp p_{j+1} | ne_G(p_j) \cap ne_G(p_{j+1}) \in I$. By local consistency, we thus have score(H') > score(G) and G is not a local optimum.

To conclude the proof, let us show by contradiction that there is no chordal graph H such that $I(G) \subsetneq I(H) \subseteq I$. Since $I(G) \subsetneq I(H)$, there exists K in the inclusion boundary of G such that $I(G) \subsetneq I(K) \subseteq I(H)$. Also, we have $\dim(K) < \dim(G)$. By consistency, we thus have score(K) > score(G) and G is not a local optimum. ∎

Note that Proposition 2 applies to all local maxima. The greedy search may thus start at any chordal graph and will return an inclusion-optimal chordal graph under the hypotheses of the proposition.

3.2 Extension to other graphical dependency models

Although we have not been able to prove it, we suspect that Proposition 2 still holds when the target dependency model I can no longer be represented perfectly by an undirected graph, but rather by a DAG.

However, as illustrated by the graphs given in Figure 2, the proposition no longer holds when I is encoded by a DAG structure with hidden variables.

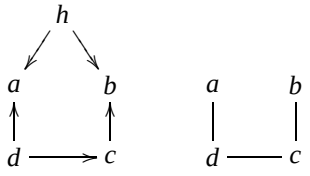


Figure 2: Suppose that I is the set of independence relations over the variables $\{a, b, c, d\}$ encoded by the DAG given in (a). Such a graph does not encode the relation $a \perp b | c$, while the chordal graph given in (b) does encode it, and is thus not inclusion-optimal for I. The neighbors of the chordal graph are obtained by adding $a - c$, adding $b - d$, removing $a - d$, removing $b - c$, or removing $c - d$. Using the local consistency property, one can see that each operation decreases the score. The chordal graph is thus a local maximum.

4 EXPERIMENTAL RESULTS

This section describes the experiments performed to assess the learning algorithm. The following settings were considered to generate simulated datasets:

- 20 or 50 binary random variables,
- a generating distribution with a chordal structure or a DAG structure.

For each setting, 30 data-generating distributions were selected with random parameters and random structure. DAG structures were drawn randomly with at most 5 parents per variable. Chordal structures were obtained by first drawing DAG structures with at most 3 parents and then chordalizing them with a greedy minimum fill-in algorithm. For each distribution, 30 independent datasets of 10^2, 10^3, 10^4 and, in the case of 20 variables, 10^5 observations were generated. For each dataset, we learned a chordal structure with the greedy search algorithm using the inclusion boundary. Also, we learned a DAG structure with the greedy search algorithm using the neighborhood obtained by legal arrow additions, removals and reversals. In both case, the BDeu scoring criterion with an equivalent sample size of 1 was used and the search was started at the empty structure. To measure KL divergences with the data-generating distribution, we estimated the parameters of the learned chordal structure, of the learned DAG structure and of the data-generating structure with the Bayesian approach corresponding to our choice of score. Then, for each dataset, the following quantities were measured:

- the dimension, i.e. number of independent parameters, of the data-generating structure,
- the dimension of the learned chordal model,
- the dimension of the learned DAG structure,
- the KL divergence from the distribution with learned parameters and learned chordal structure to the data-generating distribution,
- the KL divergence from the distribution with learned parameters and learned DAG structure to the data-generating distribution,
- the KL divergence from the distribution with learned parameters and data-generating structure to the data-generating distribution,
- in the case of chordal data-generating structure, the number of false positive lines, i.e the lines in the learned chordal structure but not in the data-generating structure, and the number of false negative lines, i.e. the lines in the data-generating structure but not the learned chordal structure.

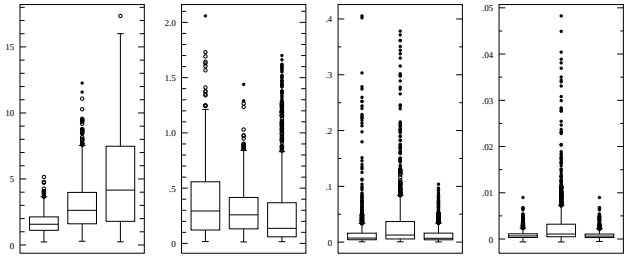
The KL divergences between a learned distribution p and a target data-generating distribution g were estimated on a dataset D of 10^4 observations drawn independently of the observations used for learning, according to the following equation:

$$KL(g \parallel p) = |D|^{-1} \sum_{i=1}^{|D|} \ln\left(\frac{P_g(X^i)}{P_p(X^i)}\right), \qquad (6)$$

where X^i denotes the ith observation of the test dataset D.

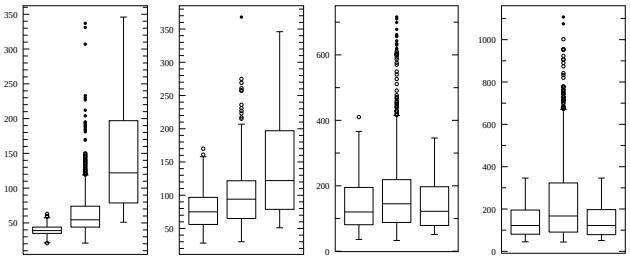
Box plots of the results are given in Figure 3 to Figure 12. They are qualitatively similar for data-generating distributions with chordal or DAG structures. Depending on the datasets sizes, one can distinguish three phases. A first phase where the learned chordal model exhibits a lower KL divergence and a lower dimension than the other models. A transition phase where the divergences and dimensions have close values. A final phase where the learned chordal model has a higher dimension and higher divergence, although the divergence tends to decrease. The first phase is expected: the model with correct structure overfits the data, while the learned model benefits from the use of a Bayesian scoring criterion that favors small structures. As the number of observations increases and we enter the third phase, the model with correct structure dominates. However, the model with learned chordal structure seems able to adapt and the difference in divergence keeps decreasing, as an expected consequence of the inclusion-optimality property.

Consider the case of data-generating distributions with chordal structures. As expected again, the number of false positive and false negative lines tends to decrease. Also, note that the number of false positives is in general much lower than the number of false negatives. This is probably due to the fact that the Bayesian score is naturally conservative and gives a high score only to independence relations that are well supported by the data.



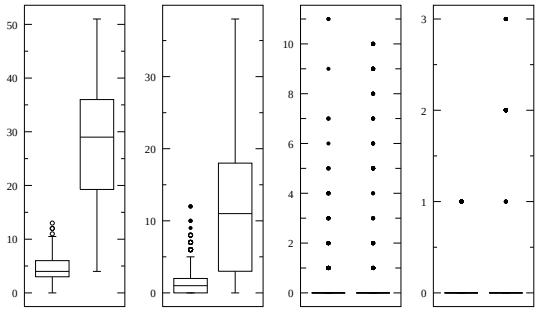
(a) $N = 10^2$ (b) $N = 10^3$ (c) $N = 10^4$ (d) $N = 10^5$

Figure 3: Estimated KL divergences to a data-generating distribution g with chordal structure over 20 binary random variables. For each sample size, the leftmost plot measures the divergence $KL(g \parallel p)$ from the distribution p with learned chordal structure and parameters, the middle plot measures the divergence $KL(g \parallel q)$ from the distribution q with learned DAG structure and parameters, and the rightmost plot measures $KL(g \parallel r)$ from the distribution r with data-generating structure and learned parameters.



(a) $N = 10^2$ (b) $N = 10^3$ (c) $N = 10^4$ (d) $N = 10^5$

Figure 4: Dimensions with 20 binary random variables and a data-generating distribution with chordal structure. For each sample size, the leftmost plot measures the dimension $d(p)$ of the learned chordal structure, the middle plot measures the dimension $d(q)$ of the learned DAG structure, and the rightmost plot measures the dimension of the data-generating structure $d(g)$.



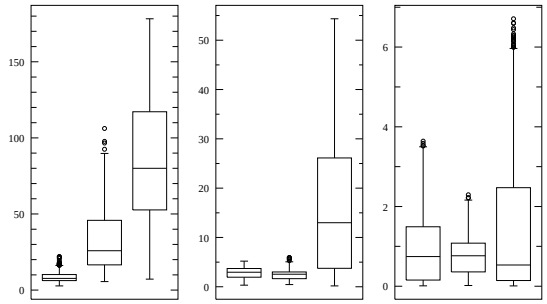
(a) $N = 10^2$ (b) $N = 10^3$ (c) $N = 10^4$ (d) $N = 10^5$

Figure 5: Number of false positive and false negative lines with 20 binary random variables and a data-generating distribution with chordal structure

5 CONCLUSION

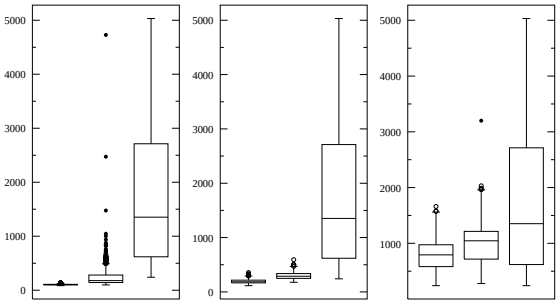
In this paper, we discussed the optimality properties of a greedy hill-climbing algorithm using the inclusion boundary neighborhood to learn the structure of a chordal graphical model. We proved that such an algorithm will asymptotically return an inclusion-optimal chordal structure if the scoring criterion is consistent and locally consistent and the dependency model of the data-generating distribution can be represented exactly by an undirected graph. Our experimental results show the practical interest of this algorithm in the context of problems where the number of variables is large and their dependency structure is sufficiently complex, be it UG-faithful or DAG-faithful.

Further theoretical work should address the extension of the above optimality property with respect to more general



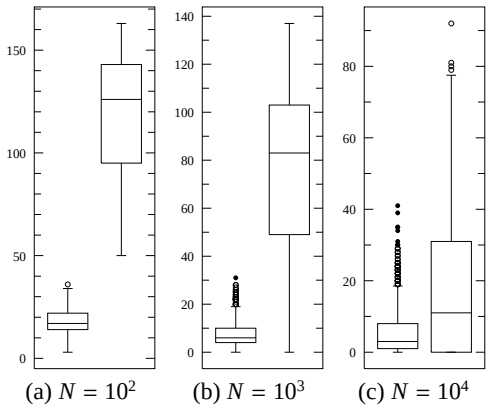
(a) $N = 10^2$ (b) $N = 10^3$ (c) $N = 10^4$

Figure 6: Estimated KL divergences with 50 binary random variables and a data-generating distribution with chordal structure (the layout is the same as in Figure 3).



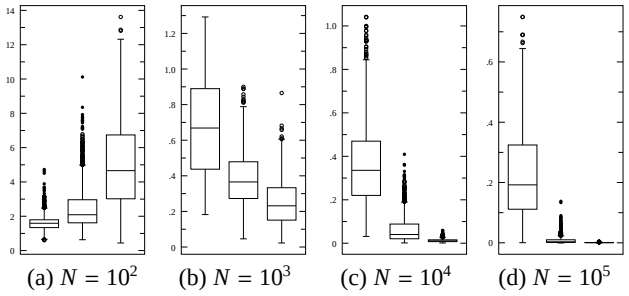
(a) $N = 10^2$ (b) $N = 10^3$ (c) $N = 10^4$

Figure 7: Dimensions with 50 binary random variables and a data-generating distribution with chordal structure (the layout is the same as in Figure 4).



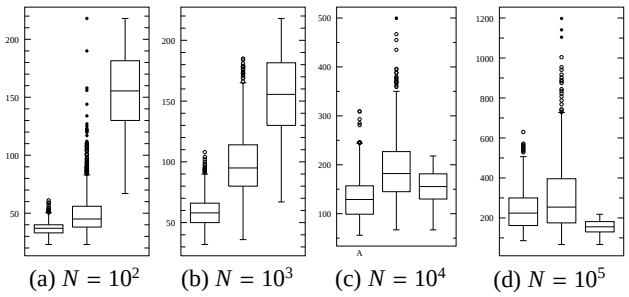
(a) $N = 10^2$ (b) $N = 10^3$ (c) $N = 10^4$

Figure 8: Number of false positive and false negative lines with 50 binary random variables and a data-generating distribution with chordal structure (the layout is the same as in Figure 5).



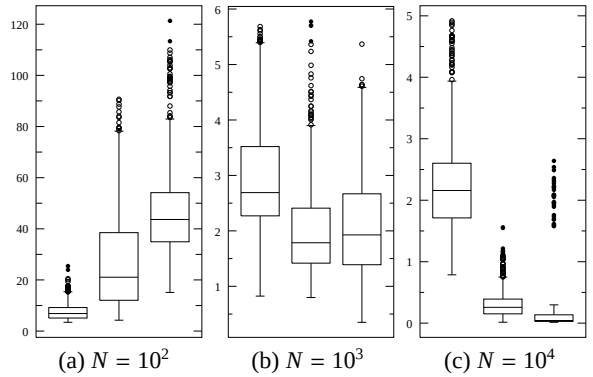
(a) $N = 10^2$ (b) $N = 10^3$ (c) $N = 10^4$ (d) $N = 10^5$

Figure 9: Estimated KL divergences with 20 binary random variables and a data-generating distribution with DAG structure (the layout is the same as in Figure 3).



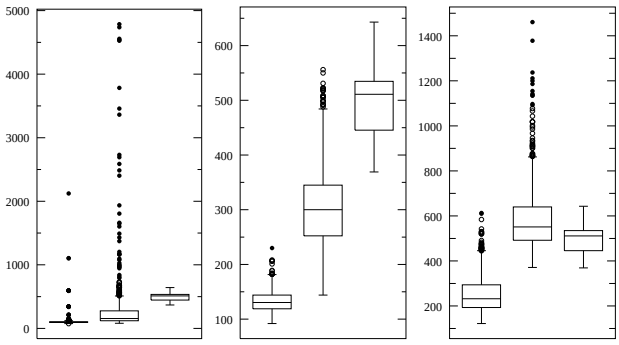
(a) $N = 10^2$ (b) $N = 10^3$ (c) $N = 10^4$ (d) $N = 10^5$

Figure 10: Dimensions with 20 binary random variables and a data-generating distribution with DAG structure (the layout is the same as in Figure 4).



(a) $N = 10^2$ (b) $N = 10^3$ (c) $N = 10^4$

Figure 11: Estimated KL divergences with 50 binary random variables and a data-generating distribution with DAG structure (the layout is the same as in Figure 3).



(a) $N = 10^2$ (b) $N = 10^3$ (c) $N = 10^4$

Figure 12: Dimensions with 50 binary random variables and a data-generating distribution with DAG structure (the layout is the same as in Figure 4).

conditions on the data-generating distributions. In particular, we believe that our optimality properties can be extended to the case where the latter is DAG faithful.

From a more practical side, it would be interesting to carry out a more in-depth empirical investigation of inclusion boundary search of chordal models with respect to inclusion boundary search in the larger space of Markov equivalence classes of DAG structures. Also, suitable combinations of these two approaches might lead to further progress for learning graphical models over large numbers of variables.

We believe also that it is of interest to further investigate the possible uses of the greedy search of inclusion optimal chordal models in the context of designing efficient approximate inference algorithms, in particular in the framework of variational approximations [10].

Acknowledgments

This work presents research results of the Belgian Network BIOMAGNET (Bioinformatics and Modeling: from Genomes to Networks), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. Vincent Auvray is supported by the "Action de recherche concertée" BIOMOD funded by the French Speaking Community of Belgium. We also want to thank Dr. R. Castelo for very fruitful discussions in September 2007, which encouraged us to direct our work on inclusion-boundary driven learning by considering the space of chordal graphs. We thank the anonymous reviewers for their very useful comments about our work.

References

[1] V. Auvray and L. Wehenkel. On the construction of the inclusion boundary neighbourhood for Markov equivalent classes of Bayesian network structures. In A. Darwiche and N. Friedman, editors, *Proceedings of 18th Conference on Uncertainty in Artificial Intelligence*, pages 26–35. Morgan Kaufmann, August 2002.

[2] R. Castelo and T. Kočka. On inclusion-driven learning of Bayesian networks. *Journal of Machine Learning Research*, 4:527–574, September 2003.

[3] D. Chickering and C. Meek. Finding optimal bayesian networks. In A. Darwiche and N. Friedman, editors, *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 94–102, San Francisco, CA, August 2002. Morgan Kaufmann.

[4] D. M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, November 2002.

[5] R. Cowell, A. Dawid, S. Lauritzen, and D. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[6] L. M. de Campos and J. F. Huete. Algorithms for learning decomposable models and chordal graphs. In D. Geiger and P. P. Shenoi, editors, *Proceedings of the 13th Conference in Uncertainty in Artificial Intelligence*, pages 46–53. Morgan Kaufmann, 1997.

[7] P. Giudici and P. J. Green. Decomposable graphical Gaussian model determination. *Biometrika*, 86(4):785–801, December 1999.

[8] R. E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, Upper Saddle River, NJ, 2003.

[9] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, 1988.

[10] W. Wiegerinck. Variational approximations between mean field theory and the junction tree algorithm. In C. Boutilier and M. Goldszmidt, editors, *Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence*, pages 626–633. Morgan Kaufmann, 2000.

APPENDIX

Proposition 3 (from [9]). *A dependency model I can be represented exactly by an undirected graph if, and only if, it satisfies the following properties:*

1. *symmetry*

$$X \perp Y|Z \in I \Leftrightarrow Y \perp X|Z \in I,$$

2. *decomposition*

$$X \perp Y \cup W|Z \in I \Rightarrow X \perp Y|Z \in I \wedge X \perp W|Z \in I,$$

3. *intersection*

$$X \perp Y|Z \cup W \in I \wedge X \perp W|Z \cup Y \in I \Rightarrow X \perp Y \cup W|Z \in I,$$

4. *strong union*

$$X \perp Y|Z \in I \Rightarrow X \perp Y|Z \cup W \in I,$$

5. *transitivity*

$$X \perp Y|Z \in I \Rightarrow X \perp \gamma|Z \in I \vee \gamma \perp Y|Z \in I,$$

where W, X, Y, and Z are disjoints subsets of vertices and γ is a singleton vertex.

Lemma 4. *Let I be a graph-isomorph dependency model. If $A_0, \ldots, A_n$ ($n \geq 3$) are sets of vertices such that $A_0 = \{x\}$, $A_n = \{y\}$, and $A_{i-1} \perp A_{i+1}|A_i \in I$ for $i = 1, \ldots, n-1$, then we have*

$$x \perp A_1 \in I \vee \cdots \vee x \perp A_{n-1} \in I \vee x \perp y|A_{n-1} \in I.$$

Proof. By transitivity and symmetry, we have

$$A_1 \perp A_3|A_2 \in I \Rightarrow x \perp A_1|A_2 \in I \vee x \perp A_3|A_2 \in I.$$

By intersection, we have

$$x \perp A_1|A_2 \in I \wedge x \perp A_2|A_1 \in I \Rightarrow x \perp A_1 \cup A_2 \in I.$$

Hence, we have

$$x \perp A_1 \cup A_2 \in I \vee x \perp A_3|A_2 \in I.$$

Repeating these steps, we obtain

$$x \perp A_1 \cup A_2 \in I \vee x \perp A_2 \cup A_3 \in I \vee \ldots \\ \vee x \perp A_{n-2} \cup A_{n-1} \in I \vee x \perp y|A_{n-1} \in I.$$

By decomposition, we thus have

$$x \perp A_1 \in I \vee \cdots \vee x \perp A_{n-1} \in I \vee x \perp y|A_{n-1} \in I.$$ ∎

Clique Matrices for Statistical Graph Decomposition and Parameterising Restricted Positive Definite Matrices

David Barber
Department of Computer Science, University College London
www.cs.ucl.ac.uk/staff/d.barber

Abstract

We introduce Clique Matrices as an alternative representation of undirected graphs, being a generalisation of the incidence matrix representation. Here we use clique matrices to decompose a graph into a set of possibly overlapping clusters, defined as well-connected subsets of vertices. The decomposition is based on a statistical description which encourages clusters to be well connected and few in number. Inference is carried out using a variational approximation. Clique matrices also play a natural role in parameterising positive definite matrices under zero constraints on elements of the matrix. We show that clique matrices can parameterise all positive definite matrices restricted according to a decomposable graph and form a structured Factor Analysis approximation in the non-decomposable case.

1 Introduction

Undirected graphs may be used to represent connectivity or adjacency structures in data. For example, in Collaborative Filtering, the nodes(vertices) in the graph may represent products, and a link(edge) between nodes i and j could be used to indicate that customers who by product i frequently also buy product j. This paper concerns decomposing the graph into well-connected clusters of nodes[1]. In Fig.1a product 3 is typically bought along with products 1 and 2, or with products 4 and 5, though these two product-groups are otherwise disjoint. A formal specification of the problem of finding a minimum number of well-connected subsets is to phrase this as `MIN CLIQUE COVER`[9, 17]. However, in some applications, provided that only a small number of links in an 'almost clique' are missing, this may be considered a sufficiently well-connected group of nodes to form a cluster. We will therefore develop a statistical technique to reveal clusters of nodes and to identify the smallest number of such clusters.

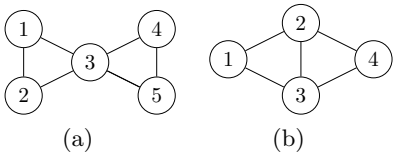


Figure 1: Two simple undirected graphs

Our main contribution is the introduction of the *clique matrix* formalism, a generalisation of the incidence matrix. We apply this to the clustering problem, in addition to demonstrating an application in constrained covariance parameterisation.

2 Clique Decomposition

The symmetric adjacency matrix has elements $A_{ij} \in \{0, 1\}$, with a 1 indicating a link between nodes i and j. For the graph in Fig.1b, the adjacency matrix is

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \tag{1}$$

where we include self connections on the diagonal. Given a graph G with adjacency matrix A, our aim is to find a 'simpler' description of A that reveals underlying cluster structure.

2.1 Two-Clique Decomposition

Given the undirected graph in Fig.1b, the *incidence matrix* Z_{inc} is an alternative description of the adjacency structure[6]. Given the V nodes in the graph,

[1] Not to be confused with graph-partitioning.

we construct Z_{inc} as follows: For each link ij in the graph, form a column of the matrix Z_{inc} with zero entries except for a 1 in the i^{th} and j^{th} row. The column ordering is arbitrary. For example, on the left

$$Z_{inc} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}, Z_{inc}Z_{inc}^{\mathsf{T}} = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 1 & 3 & 1 & 1 \\ 1 & 1 & 3 & 1 \\ 0 & 1 & 1 & 2 \end{pmatrix}$$

is an incidence matrix for the graph in Fig.1b. Taking the outer-product with itself, on the right, we see that[2]

$$A = H\left(Z_{inc}Z_{inc}^{\mathsf{T}}\right) \tag{2}$$

where $[H(M)]_{ij} = 1$ if $M_{ij} > 0$ and is 0 otherwise (*i.e.* $H(\cdot)$ is the element-wise Heaviside step function).

A useful viewpoint of the incidence matrix is that it identifies two-cliques in the graph (here we are using the term 'clique' in the non-maximal sense). There are five 2-cliques in Fig.1b, and each column of Z_{inc} specifies which elements are in each 2-clique.

2.2 Clique matrices

The incidence matrix can be generalised to describe larger cliques. Consider the following matrix as a decomposition for Fig.1b, and its outer-product:

$$Z = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad ZZ^{\mathsf{T}} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \tag{3}$$

The interpretation is that Z represents a decomposition into two 3-cliques. As in the incidence matrix, each column represents a clique, and the rows containing a '1' express which elements are in the clique defined by that column. Both Z_{inc} and Z satisfy

$$A = H\left(ZZ^{\mathsf{T}}\right) = H\left(Z_{inc}Z_{inc}^{\mathsf{T}}\right) \tag{4}$$

for Fig.1b. For clustering, Z is to be preferred against the incidence decomposition, since Z decomposes the graph into a smaller number of larger cliques. Indeed, Z solves `MIN CLIQUE COVER` for Fig.1b.

Definition 1 (Clique Matrix)**.** Given an adjacency matrix $[A]_{ij}, i, j = 1, \ldots, V$ $(A_{ii} = 1)$, a clique matrix Z has elements $Z_{i,c} \in \{0, 1\}, i = 1, \ldots, V, c = 1, \ldots C$ such that $A = H(ZZ^{\mathsf{T}})$.

A Clique Matrix $Z \in \{0, 1\}^{V \times C}$ is *minimal* for A if there exists no other clique matrix for $Z \in \{0, 1\}^{V \times C'}$ with a smaller number of columns $C' < C$.

That each column of Z expresses a clique is clear from

$$\left[ZZ^{\mathsf{T}}\right]_{ij} = \sum_k Z_{ik}Z_{jk} \tag{5}$$

For each k, the nodes i and j corresponding to $1's$ in the k^{th} column of Z give a product $Z_{ik}Z_{jk} = 1$. Since this happens for every non-zero i and j pair in the k^{th} column, all of the pair connections give rise to a non-zero product. In other words, all the nodes corresponding to non-zero elements of the k^{th} column are connected to each other, thus forming a clique.

The interpretation of the elements of ZZ^{T} is that diagonal elements $\left[ZZ^{\mathsf{T}}\right]_{ii}$ express the number of cliques/columns that vertex i occurs in. Off-diagonal elements $\left[ZZ^{\mathsf{T}}\right]_{ij}$ contain the number of cliques/columns that vertices i and j jointly inhabit.

Whilst finding a clique decomposition Z is easy (use the incidence matrix for example), finding a clique decomposition with the minimal number of columns, *i.e.* solving `MIN CLIQUE COVER`, is NP-Hard[9]. One approach would be to use a recursive procedure that searches for maximal cliques in the graph or related techniques based on finding large densely connected subgraphs[17]. The route that we take here is different and motivated by the idea that perfect clique decomposition is not necessarily desirable if the aim is only to find well-connected clusters in G.

3 Statistical Clique Decompositions

To find 'well-connected' clusters, we relax the constraint that the decomposition is in the form of cliques in the original graph. Our approach is to view the absence of links as statistical fluctuations away from a perfect clique.

Given a $V \times C$ matrix Z, we desire that the higher the overlap between rows[3] z_i and z_j is, the greater the probability of a link between i and j. This may be achieved using, for example,

$$p(i \sim j|Z) = \sigma\left(z_i z_j^{\mathsf{T}}\right) \tag{6}$$

where we define

$$\sigma(x) \equiv \left(1 + e^{\beta(0.5-x)}\right)^{-1} \tag{7}$$

and β controls the steepness of the function. The 0.5 shift in Eq. (7) ensures that σ approximates the step-function, since the argument of σ is an integer. Under Eq. (6), if z_i and z_j have at least one '1' in the same position, $z_i z_j^{\mathsf{T}} - 0.5 > 0$ and $p(i \sim j)$ is high. Absent links contribute $p(i \not\sim j|Z) = 1 - p(i \sim j|Z)$. β controls how strictly $\sigma(ZZ^{\mathsf{T}})$ matches A; for large β, very little flexibility is allowed and only cliques will be identified. For small β, subsets that would be cliques if it were not for a small number of missing links, are clustered together. The setting of β is user and problem dependent.

[2] $(\cdot)^{\mathsf{T}}$ represents matrix transpose.

[3] We use lower indices z_i to denote the the i^{th} row of Z.

Given Z, and assuming each element of the adjacency matrix is sampled independently from the generating process, the joint probability of observing A is (neglecting its diagonal elements),

$$p(A|Z) = \prod_{i\sim j} \sigma\left(z_i z_j^{\mathsf{T}}\right) \prod_{i\not\sim j} \left(1 - \sigma\left(z_i z_j^{\mathsf{T}}\right)\right)$$

The ultimate quantity of interest is the posterior,

$$p(Z|A) \propto p(A|Z)p(Z) \tag{8}$$

where $p(Z)$ is a prior over clique matrices. Later we place a prior on Z to encourage the smallest number of clusters to be identified (and hence for the size of the clusters to be large). However, since finding such Z, even in the case of a fixed desired number of clusters, C, is hard, we develop an algorithm to approximately discover clique matrices, before discussing non-uniform priors $p(Z)$.

4 Finding Z for a fixed cluster number

Formally, our task is to find the Most likely A Posteriori (MAP) solution $\arg\max_Z p(A|Z)$ where Z is a $V \times C$ binary matrix. A variety of deterministic and randomised methods could be brought to bear on this problem. The approach we take here is to approximate the marginal posterior $p(z_{ij}|A)$ and then to assign each z_{ij} to that state which maximises this posterior marginal (MPM). This has the advantage of being closely related to marginal likelihood computations, which will prove useful later for addressing the issue of finding the number of clusters. Here we develop a straightforward variational approach based on a simple factorised approximation to the posterior.

4.1 Mean Field Approximation

Given the intractable $p(Z|A) \propto p(A|Z)$, a fully factorised mean-field approximation (see, *e.g.* [20])

$$q(Z) = \prod_{i=1}^{V}\prod_{c=1}^{C} q(z_{i,c}) \tag{9}$$

can be found by minimising the KL divergence

$$KL(q,p) = \langle \log q \rangle_q - \langle \log p \rangle_q \tag{10}$$

where $\langle\cdot\rangle_q$ represents expectation with respect to q. The first 'entropic' term simply decomposes into $\sum_{i,c} \langle \log q(z_{i,c}) \rangle$. The second, 'energy' term, up to a constant is

$$\sum_{i\sim j} \left\langle \log\sigma\left(\sum_c z_{ic}z_{j,c}\right)\right\rangle_q + \sum_{i\not\sim j} \left\langle \log\left(1-\sigma\left(\sum_c z_{ic}z_{j,c}\right)\right)\right\rangle_q \tag{11}$$

The first term of Eq. (11) encourages graph links to be preserved under the decomposition, and is given by

$$\sum_{i\sim j} \left\langle f\left(\sum_{d=1}^{C} z_{id}z_{jd}\right)\right\rangle_{\prod_{e=1}^{C} q(z_{ie})q(z_{je})} \tag{12}$$

where $f(x) \equiv \log\sigma(x)$. Minimising Eq. (10) can be achieved by differentiation. Differentiating the energy contribution from the present links, Eq. (12) with respect to $q(z_{kc})$ we identify two cases: when $i = k$ and when $j = k$. Due to symmetry, the derivative is

$$2\sum_{k\sim j} \left\langle f\left(\sum_d z_{kd}z_{jd}\right)\right\rangle_{\prod_e q(z_{je})\prod_{g\neq c} q(z_{kg})} \equiv \Psi(Q) \tag{13}$$

Similarly, the derivative of the absent-links energy is

$$2\sum_{k\not\sim j} \left\langle f'\left(\sum_d z_{kd}z_{jd}\right)\right\rangle_{\prod_e q(z_{je})\prod_{g\neq c} q(z_{kg})} \equiv \Psi'(Q) \tag{14}$$

where $f'(x) \equiv \log\left(1-\sigma(x)\right)$. Equating the derivative of Eq. (10) to zero, a fixed point condition for each $q_{k,c}$ $k = 1,\ldots,V$,$c = 1,\ldots,C$ is

$$q(z_{kc}) \propto e^{\Psi(Q)+\Psi'(Q)} \tag{15}$$

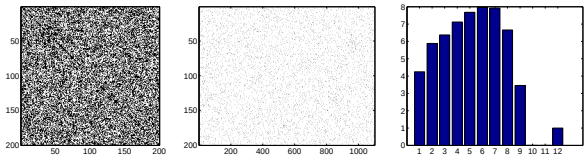
A difficulty here is that neither $\Psi(Q)$ nor $\Psi'(Q)$ are easy to compute, due to the non-linearities. A simple Gaussian Field approximation[3] assumes $\sum_d z_{kd}z_{jd}$ is Gaussian distributed for a fixed state of $z_{i,c}$. In this case, we need to find the mean and variance of $\sum_d z_{kd}z_{jd}$. Writing $\theta_{ab} \equiv q\left(z_{ab}=1\right)$, and using the independence of q, the mean is given by

$$\mu_{kj} = z_{kc}\theta_{jc} + \sum_{d\neq c} \theta_{kd}\theta_{jd}$$

A similar expression is easily obtained for the variance σ^2_{kj}. The Gaussian Field approximation then becomes,

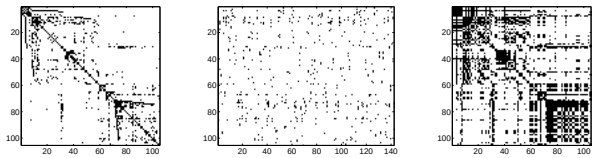
$$q(z_{kc}) \propto e^{2\left\langle \sum_{j\sim k} f(x) + \sum_{j\not\sim k} f'(x)\right\rangle_{\mathcal{N}\left(x|\mu_{kj},\sigma^2_{kj}\right)}} \tag{16}$$

where the one dimensional averages are performed numerically. By evaluating Eq. (16) for the two states



(a) (b) (c)

Figure 2: (a) Adjacency matrix for the DIMACS `brock200-2` `MAX-CLIQUE` challenge. Black denotes the presence of a link. (b) Clique Matrix. (c) Log_2-histogram of clique occurrence (+1); correctly solves `MAX-CLIQUE` (12) as well as identifying all remaining clusters.



(a) (b) (c)

Figure 3: (a) Adjacency matrix of 105 Political Books (black=1). (b) Clique matrix: 521 non-zero entries. (c) Adjacency reconstruction using an Approximate Clique Matrix with 10 cliques – see also Fig. 4.

of z_{kc} (and noting that the mean and variance of the field depends on these states), the approximate update for θ_{kc} is obtained. A simpler alternative is to assume that the variance of the field is zero, and approximate the averages by evaluating the functions at the mean of the field. We found that this latter procedure often gives satisfactory performance and therefore used this simpler and faster approach in the experiments.

One epoch corresponds to updating all the $\theta_{kc}, k = 1, \ldots, V, c = 1 \ldots, C$. During each epoch the order in which the parameters are updated is chosen randomly.

5 Finding the number of clusters

To bias the contributions to A to occur from a small number of columns of Z, we first reparameterize Z as

$$Z = \left(\alpha_1 z^1, \ldots, \alpha_{C_{max}} z^{C_{max}}\right) \tag{17}$$

where $\alpha_c \in \{0, 1\}$ play the role of indicators and z^c is the vector of column c of Z. C_{max} is an assumed maximal number of clusters we desire to find. Ideally, we would like to find a likely solution Z with a low number of indicators $\alpha_1, \ldots, \alpha_{C_{max}}$ in state 1. To achieve this we define a prior on α[4],

$$p(\alpha|\nu) = \prod_c \nu^{I[\alpha_c=1]} (1-\nu)^{I[\alpha_c=0]} \tag{18}$$

To encourage a small number of $\alpha's$ to be used, we use a Beta prior $p(\nu)$. This gives rise to a Beta-Bernoulli distribution

$$p(\alpha) = \int_\nu p(\alpha|\nu)p(\nu) = \frac{B(a+N, b+C_{max}-N)}{B(a,b)} \tag{19}$$

where $B(a, b)$ is the normalisation constant of the beta-distribution. $N \equiv \sum_{c=1}^{C_{max}} I[\alpha_c = 1]$, namely the number of indicators in state 1. To encourage strongly that a small number of components should be active, we set $a = 1, b = 3$. Through Eq. (17), the prior on α thus induces a prior on Z. The resulting distribution $p(Z, \alpha|A) \propto p(Z|\alpha)p(\alpha)$ is formally intractable.

[4] $I[x = y]$ is 1 if $x = y$ and 0 otherwise.

5.1 Variational Bayes

To deal with the intractable joint posterior we adopt a similar strategy to the fixed C case and employ a variational procedure to seek a factorised approximation $p(\alpha, Z|A) \approx q(\alpha)q(Z)$ based on minimising

$$KL(q(\alpha)q(Z), p(\alpha, Z|A)) \tag{20}$$

$q(Z)$ updates

A fixed point condition for the optimum of Eq. (20) is

$$q(Z) \propto e^{\langle \log p(A|Z,\alpha)\rangle_{q(\alpha)}} \approx e^{\log p(A|Z,\langle\alpha\rangle)} \tag{21}$$

The average over $q(\alpha)$ in Eq. (21) in the first expression is complex to carry out and we simply approximate at the average value of the distribution. This reduces the problem to one similar to that of inferring Z for a fixed C, as in Section 4.1. We therefore make the same assumption that $q(Z)$ factorizes according to Eq. (9). This gives updates of the form Eq. (16) where α has been set to its mean value.

$q(\alpha)$ updates

A fixed point condition for the optimum of Eq. (20) is

$$q(\alpha) \propto p(\alpha) e^{\langle \log p(A|Z,\alpha)\rangle_{q(Z)}},$$

Additionally we assume that $q(\alpha) = \prod_c q(\alpha_c)$. The resulting update

$$q(\alpha_c) \propto e^{\langle \log p(A|Z,\alpha)\rangle_{q(Z)} + \langle \log p(\alpha)\rangle_{\prod_{d\neq c} q(\alpha_d)}}$$

is difficult to compute and we take the naive approach of replacing averages by evaluation at the mean

$$q(\alpha_c) \propto p(\alpha_c, \langle \alpha_{\setminus c} \rangle) p(A|\langle z \rangle, \alpha_c, \langle \alpha_{\setminus c} \rangle) \tag{22}$$

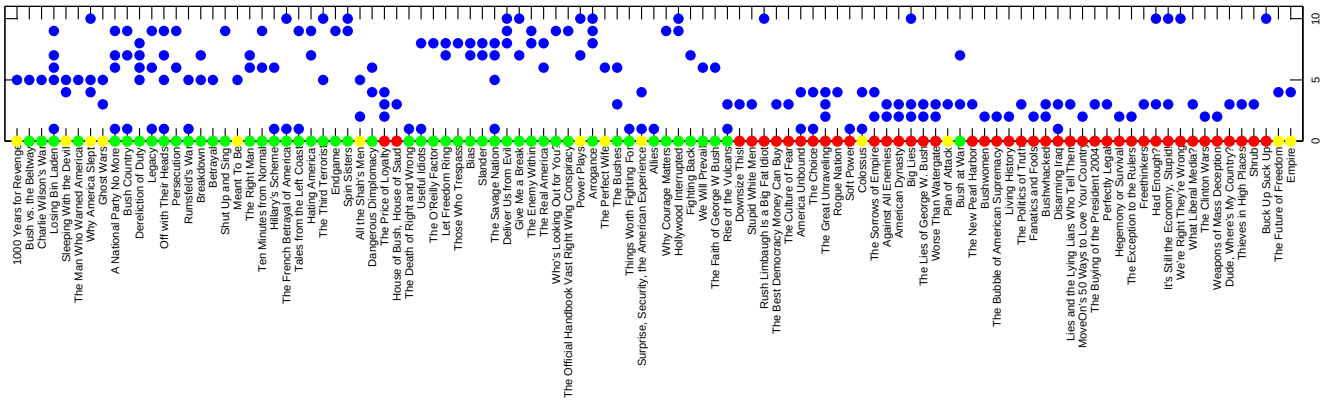


Figure 4: Political Books. Plotted is the 105×10 matrix Z, found by approximating the 105×105 adjacency co-bought matrix, where a dot indicates $q(z_{i,c}) > 0.5$. By inspection, cliques 5,6,7,8,9 largely correspond to 'conservative' books. Green indicates 'conservative' books, yellow 'neutral' and red 'liberal' books.

Since α_c is binary, we can easily find Eq. (22) by evaluating at its two states.

Formally, the prior $p(\alpha)$ requires $\alpha \in \{0,1\}^C$. However, in the above approximation, the mean α is non-binary. To deal with this, we replace $\sum_c I[\alpha_c = 1]$ by $\sum_c \langle \alpha_c \rangle$ and $\sum_c I[\alpha_c = 0]$ by $C_{max} - \sum_c \langle \alpha_c \rangle$. Since the expressions are valid for non-integer sums, this approximate procedure remains well defined.

The algorithm then updates $q(\alpha)$ and $q(Z)$ until convergence. The effect is that, beginning with C_{max} clusters, under the updating, the posterior assigns α's not required to state zero.

6 Demonstrations

DIMACS `MAX-CLIQUE`

Our method aims to find a complete characterisation of an undirected graph into constituent clusters. By setting β suitably high ($\beta = 10$ in the experiments), we impose that perfect cliques constitute clusters. In Fig.2a we show the adjacency matrix for a 200 vertex graph, taken from the DIMACS 1996 `MAX CLIQUE` challenge [4]. This graph was constructed to hide the largest clique in the graph and make it difficult to find based on the recursive algorithms of the time. Whilst more recent algorithms have been constructed that readily find the largest clique in this graph [13], this problem serves as an interesting baseline to see if our algorithm, in searching for a complete decomposition, also solves `MAX-CLIQUE` for this graph. Running our Mean-Field algorithm with $C_{max} = 2000$ results in a clique-decomposition, Fig.2b, containing 1102 cliques[5]. In Fig.2c we plot a log histogram of the cluster sizes, indicating that there is only a single largest clique of size 12. The largest clique in the graph is indeed 12[4].

Political Books Clustering

The data consists of 105 books on US politics sold by the online bookseller Amazon. Edges in graph G, Fig.3a, represent frequent co-purchasing of books by the same buyers, as indicated by the 'customers who bought this book also bought these other books' feature on Amazon[10]. Additionally, books are labelled 'liberal', 'neutral', or 'conservative' according to the judgement of a politically astute reader[6]. Running our algorithm with an initial $C_{max} = 200$ cliques, the posterior contains 142 cliques[7], Fig.3b, giving a perfect reconstruction of the adjacency A. For comparison, the incidence matrix has 441 2-cliques. To cluster the data more aggressively, we fix $C = 10$ and run our algorithm. As expected, this results only in an approximate clique decomposition, $A \approx H(ZZ^{\mathsf{T}})$, as plotted in Fig.3c. The resulting 105×10 approximate clique matrix is plotted in Fig. 4 and demonstrates how individual books are present in more than one cluster. Interestingly, the clusters found only on the basis of the adjacency matrix have some correspondence with the ascribed political leanings of each book.

7 Latent Parameterisations for Zero-Constrained Positive Matrices

We may use an undirected graph G to represent zero constraints on a positive definite matrix K. In particular, missing edges in G with adjacency $A_{ij} = 0$, correspond to zero entries $K_{ij} = 0$[8]. An example ap-

[5] This takes roughly 30s using a 1Ghz machine.

[6] See `www-personal.umich.edu/~mejn/netdata/`.

[7] This take roughly 10s on a 1GHz machine.

[8] In a Gaussian context, missing edges in G typically correspond to missing edges in the inverse covariance. Much of our initial discussion relates only to constraining positive

plication would be to fit a Gaussian to data under the constraint that specified elements of the covariance are zero. In such cases, it is useful to have a parameterisation of the allowed space of covariances. We denote the space of positive definite matrices constrained through G by $M^+(G)$. Our approach is based on the simple observation that by replacing non-zero entries of a clique matrix Z with arbitrary real values, $Z \to Z^*$, the matrix $Z^* \left(Z^*\right)^\mathsf{T}$ is positive (semi) definite. An immediate question is the richness of such a parameterisation – can all of $M^+(G)$ be reached in this way?

7.1 Decomposable Case

For G decomposable, parameterising $M^+(G)$ is straightforward[14, 12, 19]. For example one may appeal to the following:

Theorem 1 (Paulsen et al., 1989)**.** *The following are equivalent for an undirected graph G: (i) the graph is decomposable; (ii) there exists a permutation of the vertices such that with respect to this renumbering every $K \in M^+(G)$ factors as $K = T^\mathsf{T} T$ with $T \in M(G)$ and T upper triangular.*

For decomposable G, provided the vertices are perfect elimination ordered, the Cholesky factor has the same structure as G[19]. In other words, provided the vertices are ordered correctly, the lower triangular part of the adjacency matrix is a clique matrix and furthermore parameterises all of $M^+(G)$. All positive definite matrices under decomposable zero-constraints can therefore be parameterised by some clique matrix.

Definition 2 (Expanded Clique Matrix)**.** Given a Clique Matrix $Z \in \{0,1\}^{V \times C}$, the Expanded Clique matrix consists of Z appended with columns corresponding to all unique sub-columns of Z. A subcolumn of z^c is defined by replacing one or more entries containing $z_i^c = 1$ by $z_i^c = 0$.

The expanded Clique Matrix corresponding to the minimal clique matrix derived from Fig.1b is

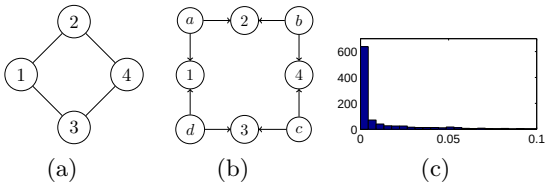
$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{23}$$

In the above, the expansion is ordered such that all 3-cliques are enumerated, then all 2-cliques and finally all 1-cliques.

Starting from a minimal clique matrix for a decomposable graph, the expansion of this minimal clique matrix must contain all the columns of the Cholesky factor T^T. For the example for G in Fig.1b, the lower triangular Cholesky factor is[9]

$$\begin{pmatrix} * & 0 & 0 & 0 \\ * & * & 0 & 0 \\ * & * & * & 0 \\ 0 & * & * & * \end{pmatrix}$$

which corresponds to columns $1, 2, 7, 11$ of the expanded clique matrix, Eq. (23). Clearly, in general, the expanded clique matrix is an over-parameterisation of $M^+(G)$ for decomposable G.



(a) (b) (c)

Figure 5: (a) Non-decomposable graph. (b) Correlations can be induced via latent variables.(c) Histogram of the rms errors in approximating covariances according to graph (a) with an expanded incidence matrix.

7.2 Non-decomposable Case

For G non-decomposable, no explicit parameterisation is generally possible and techniques based on Positive Definite matrix completion are required[12, 18, 5, 14]. For the specific example in Fig.5a, the lower Cholesky factor has the form

$$\begin{pmatrix} c_{11} & 0 & 0 & 0 \\ c_{21} & c_{22} & 0 & 0 \\ c_{31} & c_{32} & c_{33} & 0 \\ 0 & c_{42} & c_{43} & c_{44} \end{pmatrix}, \text{ with } c_{21}c_{31} + c_{22}c_{32} = 0 \tag{24}$$

which can be found explicitly in this case. However, more generally, for non-decomposable graphs, one cannot identify those elements of the Cholesky factor which may be set to zero, with the remainder determined by the positivity requirement[14, 19].

An alternative is to use latent variables to explicitly parameterise $M^+(G)$. One may use Factor Analysis[1]

$$x = F\epsilon,\ \epsilon \sim \mathcal{N}(0, I) \qquad \Rightarrow \Sigma = FF^\mathsf{T}$$

where the factor matrix F is suitably structured in order to force zeros in specific elements of Σ[10].

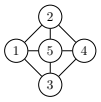
definite matrices and is independent of its application.

[9] In general, for a matrix with elements $d_{ij} \in \{0, 1\}$, we use D^* to denote a matrix with $d^*_{ij} = 0$ if $d_{ij} = 0$, and arbitrary values elsewhere.

[10] By writing $F = [\tilde{F}|D]$ where D is diagonal, this is explicitly Factor Analysis. Unlike standard FA, the matrix $\tilde{F}$ will typically be non-square and sparse.

A special case of the above is to use a latent variable to induce correlation between x_1 and x_2 via a local Directed Graph element $x_1 \leftarrow \epsilon_{12} \rightarrow x_2$. For each edge in G, a corresponding latent ϵ can thus be introduced to form correlations between all pairs of variables, without introducing correlations on missing edges in G[8]. By taking $F = [Z^*_{inc}|I^*]$, it is clear that this latent variable approach (see, for example, [16]) is reproduced and is a special case of restricting Cliques to Incidence matrices.

To show that not all of $M^+(G)$ can be reached by clique matrices, consider Fig.5a. In this particularly simple case, the minimal clique matrix is the same as the incidence matrix, and the expanded clique matrix is simply the incidence matrix with the identity matrix appended. In this case, therefore, the expanded clique matrix contains columns with only two non-zero entries. However, the Cholesky factor Eq. (24) contains columns with 3 non-zero entries, so that there is no immediate assignment of $[Z^*_{inc}|I^*]$ which will match the Cholesky factor.



For the non-decomposable graph the minimal clique matrix contains 3-cliques so that its expansion contains columns that an expansion based on an incidence matrix would not. In this case our approximate parameterisation is therefore richer than would be obtained from simply introducing a latent auxiliary variable for each edge of the graph[8, 15].

7.3 Maximum Likelihood Solution

In fitting a Gaussian $\mathcal{N}(0, \Sigma)$ to zero mean data, with sample covariance S, the ML solution minimises

$$\kappa(\Sigma) \equiv \mathrm{Tr}\left(\Sigma^{-1}S\right) + \log\det\Sigma \tag{25}$$

Our interest is to minimize κ subject to zero constraints on Σ specified through G, with $\sigma_{ij} = 0$ if $A_{ij} = 0$. For G decomposable, the problem is essentially trivial, since $M^+(G)$ is easily characterized via a structured Cholesky factor, $\Sigma \equiv C^{\mathsf{T}}(\theta)C(\theta)$ see for example [14], for which one can parameterise Eq. (25) using $\kappa(\theta)$ and perform unconstrained minimisation over the free parameters θ of the Cholesky factor.

In the non-decomposable G case, no explicit parameterisation of $M^+(G)$ is feasible. A common approach in this case is to recognise that solutions to this satisfy $\left[\Sigma^{-1}_{ij}\right] = \left[\Sigma^{-1}S\Sigma^{-1}\right]_{ij}$ for $A_{ij} = 1$ and $\sigma_{ij} =$ otherwise[2] and define iterative procedures to solve this equation[7]. Alternatively, Positive Definite Completion methods may be used to parameterise $M^+(G)$. Our approach uses the parameterisation $\Sigma = Z^*\left(Z^*\right)^{\mathsf{T}}$ where Z should be chosen as large as can be computationally afforded. Z can be determined by running the algorithm of Section 4.1[11]. Although for non-decomposable G, not all of $M^+(G)$ is guaranteed reachable through this parameterisation, one may expect that numerically this may be sufficiently close. A benefit of this approach is that one may then minimize Eq. (25) with respect to the free parameters of Z^* using any standard optimisation technique, and convergence is guaranteed. Since our parameterisation has a natural latent variable representation (it is a form of structured Factor Analysis), EM and Bayesian techniques can also be used in this case. A numerical example is plotted in Fig.5c. We take the 4×8 expanded clique matrix corresponding to Fig.5a and minimise Eq. (25) with respect to the non-zero entries of the clique matrix[12]. Each sample matrix S is generated randomly by drawing values of the Cholesky factor Eq. (24) independently from a zero mean unit variance Gaussian. In Fig.5c we plot the root mean square error between the learned Σ and sample covariance S, averaged over all non-zero components of Σ. The histogram of the error, computed from 1000 simulations shows that, whilst a few have appreciable error, the vast majority of cases are numerically well approximated by the expanded clique matrix technique, even though the graph G is non-decomposable.

8 Summary

We introduced a graph matrix decomposition based on an extension of the incidence matrix concept. Finding the clique decomposition corresponding to the smallest number of cliques is a hard problem, and we considered a relaxed version of the problem to find an approximate clique decomposition based on a variational algorithm. The approach can be seen as a form of binary factorisation of the adjacency matrix[13]. Clear extensions of this work would be to consider alternative approximate inference schemes, including sampling methods, for which 'infinite' extensions are also available[11]. An application of clique matrices is to

[11] A heuristic is to initialise Z for the variational algorithm based on the lower Cholesky factor of a spanning decomposable graph, augmented with missing two cliques. Once the algorithm converges to an approximate minimal clique matrix, its expansion is used to form the parameterisation Z^*.

[12] We chose this simple case since the exact parameterisation of all $M^+(G)$ is easy to write down. Whilst here the expanded clique and incidence matrices are equivalent, the reader should bear in mind that in more complex situations, the expansion based on a clique matrix provides a richer parameterisation than that of the incidence matrix.

[13] A parallel development to our own work is [11], which considers binary factorisation of more general matrices. Thanks to Zoubin Ghahramani for pointing me to this.

parameterising positive definite matrices under specified zero constraints. We showed that constraints corresponding to decomposable graphs trivially admit a clique matrix representation, and how our structured Factor Analysis technique can be used to approximate the non-decomposable case. This is a richer parameterisation than those latent models which consider only pairwise correlations in forming the latent model. Indeed, the so-called ancillary variable technique is a special case of using incidence, as opposed to clique matrices. The latent variable formulation additionally offers an alternative to recent works on conjugate priors for constrained covariances in Bayesian learning.

C-code for clique matrices is available from the author.

Acknowledgements

I thank R. Silva, T. Cemgil and M. Herbster for discussions. Supported in part by the European PASCAL Network of Excellence, IST-2002-506778.

References

[1] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley-Interscience, 3rd edition, 2003.

[2] T. W. Anderson and I. Olkin. Maximum-likelihood estimation of the parameters of a multivariate normal distribution. *Linear Algebra and its Applications*, 70:147–171, 1985.

[3] D. Barber and P. Sollich. Gaussian Fields for Approximate Inference in Layered Sigmoid Belief Networks. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2000.

[4] M. Brockington and J. Culberson. Camouflaging Independent Sets in Quasi-Random Graphs. In D. S. Johnson, editor, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, volume 26 of *DIMACS*. American Mathematical Society, 1996.

[5] S. Chaudhuri, M. Drton, and T S. Richardon. Estimation of a covariance matrix with zeros. *Biometrika*, 94:199–216, 2007.

[6] R. Diestel. *Graph Theory*. Springer, 2005.

[7] M. Drton and T. Richarson. A New Algorithm for Maximum Likelihood Estimation in Gaussian Graphical Models for Marginal Independence. *Uncertainty in Artificial Intelligence*, 2003.

[8] D. B. Dunsen, J. Palomo, and K. Bollen. Bayesian Structural Equation Modeling. SAMSI 2005-5, Statistical and Applied Mathematical Sciences Institute, 2005.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[10] V. Krebs. Political books data. www.orgnet.com.

[11] E. Meeds, Z. Ghahramani, R. Neal, and S. Roweis. Modeling dyadic data with binary latent factors. *Neural Information Processing Systems*, 19:977–984, 2008.

[12] V. I. Paulsen, S. C. Power, and R. R. Smith. Schur products and matrix completions. *Journal of Functional Analysis*, 85:151–178, 1989.

[13] W. J. Pullan and H. H. Hoos. Dynamic local search for the maximum clique problem. *Journal of Artificial Intelligence Research (JAIR)*, 25:159–185, 2006.

[14] A. Roverato. Hyper Inverse Wishart Distribution for Non-decomposable Graphs and its Application to Bayesian Inference for Gaussian Graphical Models. *Scandanavian Journal of Statistics*, 29:391–411, 2002.

[15] R. Silva, W. Chu, and Z. Ghahramani. Hidden Common Cause Relations in Relational Learning. *Neural Information Processing Systems*, 2007.

[16] R. Silva and Z. Ghahramani. Bayesian Inference for Gaussian Mixed Graph Models. *Uncertainty in Artificial Intelligence*, 2006.

[17] S. S. Skiena. *The algorithm design manual*. Springer-Verlag, New York, USA, 1998.

[18] T. P. Speed and H. Kiiveri. Gaussian markov distributions over finite graphs. *Annals of Statistics*, 14:138–150, 1986.

[19] N. Wermuth. Linear Recursive Equations, Covariance Selection, and Path Analysis. *Journal of the American Statistical Association*, 75(372):963–972, 1980.

[20] W. Wiegerinck. Variational approximations between mean field theory and the junction tree algorithm. *Uncertainty in Artificial Intelligence*, 16:626–633, 2000.

Sensitivity analysis in decision circuits

Debarun Bhattacharjya and Ross D. Shachter
Department of Management Science and Engineering
Stanford University
Stanford, CA 94305, USA
E-mail: debarunb@stanford.edu, shachter@stanford.edu

Abstract

Decision circuits have been developed to perform efficient evaluation of influence diagrams [Bhattacharjya and Shachter, 2007], building on the advances in arithmetic circuits for belief network inference [Darwiche, 2003]. In the process of model building and analysis, we perform sensitivity analysis to understand how the optimal solution changes in response to changes in the model. When sequential decision problems under uncertainty are represented as decision circuits, we can exploit the efficient solution process embodied in the decision circuit and the wealth of derivative information available to compute the value of information for the uncertainties in the problem and the effects of changes to model parameters on the value and the optimal strategy.

1 INTRODUCTION

Influence diagrams are powerful communication tools and computational aids for the analysis of practical decision problems [Howard and Matheson, 1984]. Decision circuits are a recent graphical representation that have been introduced for the efficient evaluation of influence diagrams [Bhattacharjya and Shachter, 2007]. In this paper, we show that they are also useful for efficient sensitivity analysis in influence diagrams.

The phrase *sensitivity analysis* refers, in general, to understanding how the output for a system varies as a result of changes in the system's input(s). For influence diagrams, one may be concerned about how the optimal solution and the certain equivalent (CE) change with respect to a change in the parameters, i.e. the probabilities and the utilities, or a change in the informational assumptions of the problem. There are many questions that we can use sensitivity analysis to answer. For example, suppose we vary one parameter keeping all other parameters constant. For what range of this varying parameter is the current optimal strategy still optimal? How does the value change as this parameter is varied? What if the structure of the influence diagram changes and an uncertainty that would not have been revealed before we make a decision is now observed before the first decision is made? These are only some of the queries on the model that we seek to answer.

Several issues in sensitivity analysis of Bayesian belief networks have been studied, using arithmetic circuits [Darwiche, 2003] for efficient solutions [Chan and Darwiche, 2002; 2004; 2006]. Our work builds on this body of research. Arithmetic circuits are graphical representations that have been shown to be efficient at performing inference on belief networks. Decision circuits promise similar benefits in the context of sequential decision problems.

In sections 2, 3 and 4 we briefly discuss some preliminaries, and review literature on circuits and sensitivity analysis in influence diagrams. In section 5, we introduce key ideas of performing sensitivity analysis with decision circuits for normal form influence diagrams, i.e. influence diagrams involving a single decision node with no parents. We show how to perform some basic sensitivity analysis with decision circuits, such as: plotting the certain equivalent in a one-way sensitivity analysis, finding the range of a parameter over which the current optimal stays optimal, and computing the value of information of uncertainties that are not affected by decisions, using partial derivatives. This serves as an introduction to section 6, where we present results for influence diagrams that may contain multiple decision nodes. We show the challenges and necessary modifications from the previous section. Finally, section 7 describes our conclusions and directions for future work.

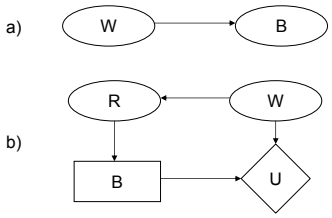


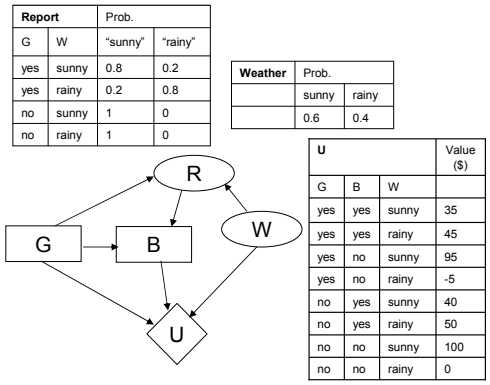
Figure 1: (a) A belief network; (b) An influence diagram.

2 PRELIMINARIES

We assume that the reader is familiar with graphical models such as Bayesian belief networks and influence diagrams (see Shachter (2007) for an overview). Consider the two examples shown in Figure 1. Figure 1a presents a belief network with two nodes, labelled W (Weather) and B (Bring umbrella). We are interested in knowing whether a friend will bring an umbrella, and we believe that it is easier to model this if we condition on the weather. Figure 1b shows the influence diagram for a decision problem in which our friend chooses whether to bring an umbrella based on her belief about the weather and her preferences, represented by the node U (Utility). She will observe a weather Report (R) before she makes her decision. We use these examples to demonstrate the concepts in later sections.

We make a distinction between *extensive* and *normal* form influence diagrams. When there is only one decision node and it has no parents, the diagram is said to be in normal form [Savage, 1954; Raiffa, 1968]. We extend that definition to allow for evidence, i.e. that we observe certain variables taking on specific values. Alternatively, when decisions are represented by separate nodes, or when the diagram has a decision node with at least one parent, the diagram is said to be in extensive form. There can be a large number of *strategies* in an influence diagram, one for each possible combination of observed uncertainties and decision alternatives. Although it is possible to convert any extensive form influence diagram into a normal form influence diagram, it may not be efficient to do so. The influence diagram shown in Figure 1b is in extensive form because the decision node B has a parent R.

In this paper we assume that the influence diagram has a single value node, referred to as utility node U. When making decisions, we choose the alternative that maximizes the probability that utility variable $U = 1$. The parents of utility node U, $\mathbf{pa}(\mathbf{U})$, are called the *value attributes* and we assess $P(U = 1|\mathbf{pa}(\mathbf{U})) = u(v(\mathbf{pa}(\mathbf{U})))$ where $u(.)$ is a von Neumann-Morgenstern utility function [von Neumann and Morgenstern, 1947] such that $U = 1$ is at least as good and $U = 0$ is at least as bad as anything that can happen, and $v(.)$ characterizes the value of the attributes in terms of a single numeraire, which we assume is dollars. Therefore, the *certain equivalent* (CE) of an uncertain V, given by $u^{-1}(E[u(V)])$, represents the certain payment that the decision-maker finds indifferent to V. Our sensitivity results will be expressed in terms of the certain equivalent because the utility values used for the internal computations have no intrinsic meaning. We also assume that the utility function $u(.)$ is strictly increasing and continuously differentiable. The most common utility functions are *linear*, $u(v) = av + b$, and *exponential*, $u(v) = -ae^{-v/\rho} + b$, where $a > 0$ and $\rho > 0$, both of which allow us to express the value of information exactly in closed form.



Report		Prob.	
G	W	"sunny"	"rainy"
yes	sunny	0.8	0.2
yes	rainy	0.2	0.8
no	sunny	1	0
no	rainy	1	0

Weather	Prob.	
	sunny	rainy
	0.6	0.4

U			Value ($)
G	B	W	
yes	yes	sunny	35
yes	yes	rainy	45
yes	no	sunny	95
yes	no	rainy	-5
no	yes	sunny	40
no	yes	rainy	50
no	no	sunny	100
no	no	rainy	0

Figure 2: An influence diagram example with numbers.

We will present some sensitivity analysis results for extensive form influence diagrams with the help of the influence diagram shown in Figure 2. In this example, our friend will decide whether to Gather evidence (G) and purchase a weather report. Her information gathering decision and the report will be known to her when she decides whether to bring her umbrella. If our friend does not gather evidence, the report is not informative and always states "sunny". The conditional probability tables and the value function $v(.)$ are shown in the figure. We assume that the decision maker has an exponential utility function $u(.)$ with risk tolerance $\rho = .02$. The optimal strategy is to gather evidence and bring the umbrella when the report says "rainy" and not to bring the umbrella when the report says "sunny". The CE of the decision problem is \$52.5.

3 ARITHMETIC AND DECISION CIRCUITS

We review basic concepts regarding arithmetic and decision circuits in this section. Throughout this paper,

variables are denoted by upper-case letters (X) and their values by lower-case letters (x). A bold-faced letter indicates a set of variables. If X is a variable with parents $\mathbf{Pa}(\mathbf{X})$, then $X\mathbf{Pa}(\mathbf{X})$ is called the family for variable X. The values of a binary variable X are denoted x and $\bar{x}$.

3.1 Arithmetic Circuits

Belief networks are associated with a unique multi-linear function over two kinds of variables, *evidence indicators* and *network parameters*. An evidence indicator λ_x is a binary (0-1) variable, with $\lambda_x = 0$ whenever X has been observed taking another value, i.e. it is observed not to be x. There is an evidence indicator associated with each possible instantiation x of each network variable X. A network parameter $\theta_{x|\mathbf{pa}(\mathbf{X})}$ represents a conditional probability, $\theta_{x|\mathbf{pa}(\mathbf{X})} = P(x|\mathbf{pa}(\mathbf{X}))$, and there is a network parameter for each possible instantation $x\mathbf{pa}(\mathbf{X})$ of family $X\mathbf{Pa}(\mathbf{X})$. Each *term* in the multi-linear function corresponds to one instantiation $\mathbf{z}$ of all the network variables $\mathbf{Z}$, involving the product of all evidence indicators and network parameters consistent with $\mathbf{z}$. The multi-linear function for a belief network is given as $f = \sum_{\mathbf{z}} \prod_{x\mathbf{pa}(\mathbf{X})\sim\mathbf{z}} \lambda_x \theta_{x|\mathbf{pa}(\mathbf{X})}$ where the sum is over every instantiation of all variables in the network and $x\mathbf{pa}(\mathbf{X}) \sim \mathbf{z}$ represents all families consistent with $\mathbf{z}$. For example, consider the belief network of Figure 1a. Suppose that W and B are binary variables with states w and $\bar{w}$, and b and $\bar{b}$ respectively. The multi-linear function for this network is: $f = \lambda_w\lambda_b\theta_w\theta_{b|w} + \lambda_{\bar{w}}\lambda_b\theta_{\bar{w}}\theta_{b|\bar{w}} + \lambda_w\lambda_{\bar{b}}\theta_w\theta_{\bar{b}|w} + \lambda_{\bar{w}}\lambda_{\bar{b}}\theta_{\bar{w}}\theta_{\bar{b}|\bar{w}}$.

The multi-linear function is a useful construct for answering inference queries in belief networks. By setting the evidence indicators to 0 or 1, we can find the probability of observing any set of network variables $\mathbf{E}$. For instance, if we assign evidence to be $\mathbf{e} = \bar{b}$ by setting $\lambda_b = 0$ and all the other three evidence indicators as 1, the function returns $P(\bar{b}) = \theta_w\theta_{\bar{b}|w} + \theta_{\bar{w}}\theta_{\bar{b}|\bar{w}}$. In general, the evidence indicators help in summing the appropriate entries in the joint probability distribution table for computing the probability of the evidence, $P(\mathbf{e})$. Furthermore, the partial derivatives of the multi-linear function also provide solutions to several common probabilistic inference queries. We list two lemmas with important relationships between inference queries and the multi-linear function, as proven in Darwiche (2003):

Lemma 1. *For evidence* $\mathbf{e}$*, we have:* $P(\mathbf{e}) = f(\mathbf{e})$.

Lemma 2. *For every variable* X *and evidence* $\mathbf{e}$ *such that* $X \notin \mathbf{E}$*, we have:* $P(x, \mathbf{e}) = \frac{\partial f}{\partial \lambda_x}(\mathbf{e})$.

Arithmetic circuits are graphical structures that efficiently represent, evaluate, and differentiate multi-linear functions. An arithmetic circuit is a rooted, directed acyclic graph whose leaf nodes are constants or variables and all other nodes represent either summation or multiplication. The *size* of an arithmetic circuit is the number of edges it contains.

The value of the multi-linear function is computed at the root of the circuit by *evaluating* the circuit in an *upward pass*, starting from the leaves and ending at the root. The result is denoted as $f(\mathbf{e})$, where $f(\mathbf{e}) = P(\mathbf{e})$ (see Lemma 1). We can calculate partial derivatives by *differentiating* the circuit through a subsequent *downward pass*, in which the parents are visited before the children. The upward and downward passes are also referred to as *sweeps*. For further details, please see Darwiche (2003).

Compact arithmetic circuits have been devised for belief networks that had previously been intractable [Darwiche, 2002; Chavira and Darwiche, 2005]. The circuit is compiled offline, where both the local structure (in the form of determinism and context-specific independence) as well as the conditional independencies of the graph at the global level are exploited. Several inference queries on the network can then be processed online, through subsequent operations on the compiled arithmetic circuit. Efficient sensitivity analysis of the belief network is possible with sweeps of the compiled circuit.

3.2 Decision Circuits

Decision circuits are arithmetic circuits augmented with maximization nodes. They represent the dynamic programming function corresponding to a sequential decision problem. The *size* of a decision circuit is the number of edges it contains. Figure 3 presents a decision circuit corresponding to the influence diagram shown in Figure 1b.

Decision circuits for influence diagrams can be constructed in the variable elimination order [Bhattacharjya and Shachter, 2007], similar to a construction technique for arithmetic circuits [Darwiche, 2000]. The evidence indicator λ_d for decision D is initialized to 0 only if the alternative d is no longer available to the decision maker. The network parameter $\theta_{d|\mathbf{pa}(\mathbf{D})}$ for a decision D is initialized to 1 if the alternative is conditionally available under scenario $\mathbf{pa}(\mathbf{D})$, and 0 otherwise. Once compiled, the decision circuit can be evaluated in an upward sweep analogous to evaluation in arithmetic circuits. Decision circuits are evaluated with evidence $\mathbf{e}' = \mathbf{e} \cup \{U = 1\}$ when $\mathbf{e}$ is observed. The best outcome $U = 1$ is also deemed to be observed since this is an MEU problem and therefore the goal is to find optimal policies that maximize the probability of the best outcome given the evidence. The value

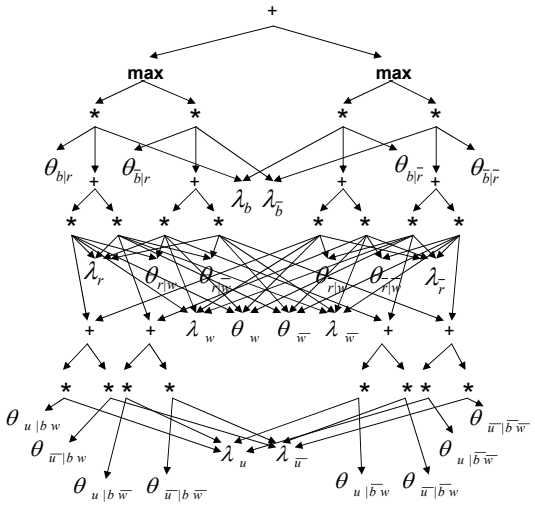


Figure 3: Decision circuit for the influence diagram shown in Figure 1b.

of the root node of the circuit is denoted $g(\mathbf{e}')$. The optimal strategy is computed on the upward sweep at the maximization nodes, where the alternative d^* with the highest value is chosen, breaking ties arbitrarily. The network parameter $\theta_{d|\mathbf{u}}$ is set to 0 for all other alternatives d. The circuit can be differentiated in a subsequent downward sweep, by treating the maximization nodes as summation nodes. Although optimal policies are determined on the upward sweep, the MEU and CE are calculated also using results from the downward sweep. Specifically, from Bhattacharjya and Shachter (2007):

Lemma 3. $MEU = \frac{g(\mathbf{e}')}{\frac{\partial g}{\partial \lambda_u}(\mathbf{e}') + \frac{\partial g}{\partial \lambda_{\bar{u}}}(\mathbf{e}')}$.

Lemma 4. *For utility function $u(.)$, we have:* $CE = u^{-1}(MEU)$.

4 SENSITIVITY ANALYSIS IN INFLUENCE DIAGRAMS

The conditional probabilities in an influence diagram can be difficult to assess due to the paucity of data, expert judgments about key uncertainties, the decision maker's imprecision regarding preferences, and several other practical reasons. As a result, it is often beneficial to inspect the change in the outputs of the decision model based on variations in the inputs of the model. Such issues fall under the umbrella of sensitivity analysis.

Sensitivity analysis has been an essential aspect of decision analysis throughout the field's development. Sensitivity analysis aids in identifying the model's critical elements, forming the basis for iterative refinement of the model, and can also be used after the analysis for defending a particular strategy to the decision maker [Howard, 1983]. Sensitivity plots displaying the certain equivalents of different strategies can help identify the important variables. Sensitivity analysis for decision problems can be broadly classified in terms of whether one parameter is varied while others are kept constant (*one-way sensitivity analysis*) or when multiple parameters are simultaneously varied (*n-way sensitivity analysis*). One-way sensitivity analysis throws light upon the critical model variables whereas n-way sensitivity analysis provides insights into the general robustness of the model.

Sensitivity analysis in belief networks explores the sensitivity of inference queries such as the probability of evidence and conditional marginal probabilities given the evidence, to the conditional probabilities of the network [Laskey, 1995; Castillo et al, 1997; Kjaerulff and van der Gaag, 2000; van der Gaag and Renooij, 2001]. Sensitivity to inference queries using arithmetic circuits has also been studied [Chan and Darwiche, 2002; 2004] as has the sensitivity of Most Probable Explanations to parameter changes [Chan and Darwiche, 2006]. Our work differs from this line of research in that it uses decision circuits to determine sensitivity of the optimal strategy and the certain equivalent to parameter changes in decision problems.

To perform sensitivity analysis for decision problems, we will vary the model output with respect to *meta-parameters*, similar to Chan and Darwiche (2002) and Nielsen and Jensen (2003). For a variable X, we analyze sensitivity to all parameters of the form $\theta_{x|\mathbf{pa}(\mathbf{X})}$ as linear functions of a meta-parameter τ. For example if X is a binary variable with no parents, we can set $\theta_x = \tau$ and $\theta_{\bar{x}} = 1 - \tau$. In this paper we assume that there are K meta-parameters, denoted $\tau_k, k = 1, ..., K$, each between 0 and 1, and that these meta-parameters are drawn explicitly in the decision circuit, although it is also possible to allow the meta-parameters to be implicit [Chan and Darwiche, 2002]. We assume that each meta-parameter is associated with only one variable. This ensures that the root value of the decision circuit is a piecewise-linear function of each meta-parameter. This can be extended to cases where the model inputs are non-linear functions of the meta-parameters, such as sensitivity to risk tolerance.

5 ANALYZING NORMAL FORM INFLUENCE DIAGRAMS

When there is only one decision node in an influence diagram and it has no parents, the diagram is said to be in normal form. We extend this definition to allow the observation that $\mathbf{E} = \mathbf{e}$. We discuss sensitivity analysis for normal form influence diagrams in this

section.

5.1 Partial derivatives

Consider an influence diagram with a single decision node D that has no parents. Let X be an arbitrary uncertain variable, with parents $\mathbf{Pa}$ (Note that D can be a parent of X). The dynamic programming function corresponding to this normal form influence diagram is given by $g = \max_d \theta_d \lambda_d \sum_{\mathbf{z}} \prod_{x,\mathbf{pa}(\mathbf{X})\sim\mathbf{z}} \theta_{x|\mathbf{pa}(\mathbf{X})}\lambda_x$ where d is an alternative for decision node D. In normal form, each alternative represents a strategy. Let EU_d be the expected utility for strategy d. Then $MEU = \max_d EU_d$, optimal strategy $d^* = argmax_d EU_d$ and $CE = u^{-1}(MEU)$. We will discuss many of the normal form sensitivity results using the following theorem, which outlines the significance of derivatives of $g(\mathbf{e}')$ and $g(\mathbf{e})$. These are the root values of the decision circuit evaluated at evidence $\mathbf{e}'$ and $\mathbf{e}$, respectively.

Theorem 1. *If evidence $\mathbf{e}' = \mathbf{e} \cup \{U = 1\}$ and $\mathbf{e}$ are swept through a decision circuit constructed for a normal form influence diagram, then for any node v:*
(i) $\frac{\partial EU_d}{\partial v} = \frac{\partial}{\partial v}\left[\frac{\frac{\partial g(\mathbf{e}')}{\partial \theta_d}}{g(\mathbf{e})}\right] = \frac{g(\mathbf{e})\frac{\partial^2 g(\mathbf{e}')}{\partial \theta_d \partial v} - \frac{\partial g(\mathbf{e})}{\partial v}\frac{\partial g(\mathbf{e}')}{\partial \theta_d}}{(g(\mathbf{e}))^2}$.
(ii) If MEU^{PI} is the maximal expected utility with perfect information on uncertainty X, then: $MEU^{PI} = \sum_x \max_d \frac{\partial EU_d}{\partial \lambda_x}$.
(iii) $\frac{\partial CE_d}{\partial v} = \frac{\frac{\partial EU_d}{\partial v}}{u'(CE_d)}$.

Proof. (i): On the downward sweep the maximization nodes are treated as summation nodes, therefore $g = \sum_d \theta_d \lambda_d \sum_{\mathbf{z}} \prod_{x,\mathbf{pa}(\mathbf{X})\sim\mathbf{z}} \theta_{x|\mathbf{pa}(\mathbf{X})}\lambda_x$ for the downward sweep. Differentiating g with respect to θ_d and evaluating at evidence $\mathbf{e}'$ results in $P(U = 1, \mathbf{e}|d)$ since only terms associated with alternative d remain in the summation. The expected utility of alternative d is $P(U = 1|\mathbf{e}, d)$, therefore $EU_d = \frac{1}{P(\mathbf{e})}\frac{\partial g(\mathbf{e}')}{\partial \theta_d}$. The result follows by recognizing that $P(\mathbf{e}) = g(\mathbf{e})$ ($\mathbf{e}$ is not responsive to any strategy d), and differentiating with respect to node v using the quotient rule of differentiation.
(ii): $MEU^{PI} = \sum_x P(x|\mathbf{e}) \max_d P(U = 1|\mathbf{e}, x, d)$
$= \sum_x \max_d P(U = 1, x|\mathbf{e}, d)$
$= \sum_x \max_d \frac{\partial EU_d}{\partial \lambda_x}$.
We have used some results from the proof of part (i), and the fact that the uncertain variable X is not affected by the decision, thus $P(x|\mathbf{e}) = P(x|\mathbf{e}, d) \forall d$.
(iii) $EU_d = u(CE_d)$; the result follows from differentiating both sides with respect to v using the chain rule of differentiation. □

Theorem 1 presents a recipe for computing $\frac{\partial EU_d}{\partial v}$ and $\frac{\partial CE_d}{\partial v}$ for any node v in the circuit, using $g(\mathbf{e})$, $g(\mathbf{e}')$ and their derivatives. The single and double derivatives of the form used in Theorem 1 can be obtained in time linear in the size of the circuit [Darwiche, 2000]. The double derivatives $\frac{\partial^2 g(\mathbf{e}')}{\partial \theta_d \partial v}$ are computed by sweeping $\frac{\partial g(\mathbf{e}')}{\partial \theta_d}$, for all strategies d, down the circuit. Thus the time complexity for obtaining $\frac{\partial EU_d}{\partial v}$ for all nodes is $O((N_S)(dc))$ where N_S is the number of strategies and dc is the size of the decision circuit. Note that the slope of CE_d with respect to v is not constant unless the utility function is linear; it will depend on the value of CE_d in general.

5.2 One-way sensitivity analysis plots

In this sub-section, we show how to create a graph of the certain equivalent for the decision problem as a function of a particular meta-parameter when all others are kept at their reference values.

The expected utility for a particular strategy is a multi-linear function of all the meta-parameters. Therefore, the expected utility for a particular strategy d is a linear function of a particular meta-parameter τ_k, keeping all other meta-parameters at their reference values. We denote this linear function as $EU_d = \alpha_d(\tau_k)$. We already have a point on this line from the initial sweep used to evaluate the circuit, which we denote as (τ_k^0, α_d^0). We also have the slope of this line, since it equals the partial derivative from part (i) of Theorem 1 choosing $v = \tau_k$. We denote the slope as $\alpha'_{d,k}$. Plotting the decision problem's CE with respect to changes in τ_k entails applying the inverse utility function $u^{-1}(.)$ to the maximum of the lines for all strategies. If the required resolution is ϵ, then the number of points required between 0 and 1 for the plot is $1/\epsilon$. The time complexity for preparing the plot for all meta-parameters, once we have the lines for expected utilities of all strategies with respect to all meta-parameters, is $O((\frac{1}{\epsilon})(K)(N_S))$ where K is the number of meta-parameters and N_S is the number of strategies.

Another approach to plotting the CE against a meta-parameter is the standard method of *sample points*, where the decision problem is re-evaluated for every sample point over a range. If the required resolution is ϵ, then the time complexity of this approach for obtaining plots for all meta-parameters is $O((\frac{1}{\epsilon})(K)(dc))$.

5.3 Admissible intervals

Another sensitivity question in the spirit of one-way sensitivity analysis is the following one. Suppose a certain meta-parameter is allowed to vary, keeping all other meta-parameters constant at their reference values. What is the range of this meta-parameter over

which the current optimal strategy remains optimal? We call this range the *admissible interval* for a meta-parameter. In other words, we investigate how robust the optimal strategy is, with respect to changes in any meta-parameter.

The admissible intervals are easy to obtain for normal form influence diagrams, based on the ideas from the previous subsection. Since we have the lines for the expected utilities of all strategies with respect to all meta-parameters, we can obtain the admissible intervals I_k for all meta-parameters simultaneously. Suppose $\Delta+$ and $\Delta-$ are the admissible positive and negative changes to τ_k from its reference value τ_k^0 such that d^* stays optimal. We present the following theorem, without proof, for computing $\Delta+$, $\Delta-$ and the intervals I_k. The proof entails finding the points at which another strategy overtakes d^* and recognising that τ_k lies between 0 and 1. The time complexity for computing these intervals for all meta-parameters, once we have the lines for expected utilities of all strategies with respect to all meta-parameters, is $O((K)(N_S))$.

Theorem 2. *For meta-parameter τ_k, the admissible positive change $\Delta+ = \min_{ds.t.\alpha'_{d,k}>\alpha'_{d^*,k}} \left[\frac{\alpha^0_{d^*}-\alpha^0_d}{\alpha'_{d,k}-\alpha'_{d^*,k}}\right]$, the admissible negative change $\Delta- = \max_{ds.t.\alpha'_{d,k}<\alpha'_{d^*,k}} \left[\frac{\alpha^0_{d^*}-\alpha^0_d}{\alpha'_{d,k}-\alpha'_{d^*,k}}\right]$, and the admissible interval $I_k = \left[\max\left(0, \tau_k^0 + \Delta+\right), \min\left(1, \tau_k^0 + \Delta-\right)\right]$.*

Binary search is another possible approach for finding the admissible intervals. The admissible interval for any meta-parameter τ_k is a convex set due to the linearity of the expected utilities for all strategies with respect to τ_k. We can therefore locate the end-points of this interval by re-evaluating the circuit at points chosen by binary search to check if the current optimal strategy is still optimal. If the required resolution is ϵ, then the number of sample points needed for the binary search is $O(-\ln(\epsilon))$. Thus the time complexity of obtaining admissible intervals for all meta-parameters by this method is $O((-\ln(\epsilon)(K)(dc))$.

5.4 Value of information

The value of information for a particular uncertainty is a useful sensitivity analysis query in sequential decision problems, specifying the maximum that the decision maker should be willing to pay to observe the uncertainty before making the first decision [Howard, 1966; Raiffa, 1968]. The double derivatives from part (i) of Theorem 1 can also compute the value of information for all uncertainties that are not affected by the decision, using part (ii) of Theorem 1 (choose $v = \lambda_x$). Here MEU^{PI} is the maximal expected utility with perfect information on uncertain variable X. If the decision maker's utility function $u(.)$ is linear or exponential, then the value of information of the uncertain variable X is the increase in the certain equivalent $u^{-1}(MEU^{PI}) - u^{-1}(MEU)$. In general, this is usually a good approximation for the value of information, even if the decision maker's utility function has a form other than linear or exponential [Raiffa, 1968].

Once we have the results from Theorem 1, computing the value of information involves summing and maximizing the partial derivatives. If the number of variables analyzed for value of information is $\sharp var$, and assuming that the maximum number of possibilities for all of these variables is bounded by some constant, then the time complexity for obtaining the value of information for all these variables is $O((N_S)(\sharp var))$.

6 ANALYZING EXTENSIVE FORM INFLUENCE DIAGRAMS

Sensitivity analysis in normal form influence diagrams is relatively easy to describe because we explicitly represent all the strategies. Here we discuss techniques for analyzing extensive form influence diagrams. All of these methods can also apply to normal form diagrams. We assume the standard conditions in the influence diagram literature such as "no forgetting" [Howard and Matheson, 1984; Shachter, 1986].

6.1 One-way sensitivity analysis plots

The sample points method from the previous section directly applies for extensive form influence diagrams as well. It is also possible to plot the CE for a particular strategy s against τ_k using decision circuits. To do this, the network parameters for decisions $\theta_{d|\mathbf{pa(D)}}$ need to be set according to s. Note that these have been preset according to the current optimal strategy s^* after the initial upward sweep to evaluate the optimal strategy, hence they need to be reset if $s \neq s^*$. Once the network parameters for decisions are chosen appropriately, we can sweep up and down the decision circuit, treating the maximization nodes as summation nodes on both sweeps.

Figure 4 is a one-way sensitivity analysis plot for the influence diagram shown in Figure 2, with respect to meta-parameter τ_1, the probability that the weather is sunny. The plot displays the variation in the CE of the decision problem as well as the CE of the current optimal strategy s^* with respect to changes in τ_1. We observe that for most values of the meta-parameter, the CE for the decision problem does not vary too much, and thus the model is robust to small changes in τ_1 around its current value of 0.6. The figure supports the intuition that for extreme values of the probability of sunshine, it is optimal to decide whether to bring

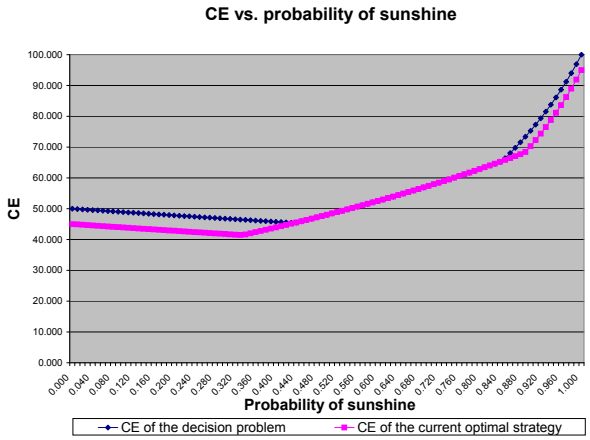


Figure 4: One way sensitivity analysis plot for an example.

the umbrella or not without paying for the evidence.

6.2 Admissible intervals

In this section, as an extension of our discussion in the section on normal form influence diagrams, we present the admissible interval algorithm, which returns bounds on the range over a particular meta-parameter for which the optimal strategy s^* remains optimal when all other meta-parameters are kept at their reference values. The algorithm finds these bounds for all meta-parameters simultaneously. We assume that the optimal strategy s^* and the MEU have already been evaluated from initial sweeps through the decision circuit. The algorithm computes partial derivatives from a downward sweep starting from every maximization node. Before we describe the algorithm, we distinguish between *active* and *inactive max nodes* in the decision circuit. Inactive max nodes are those maximization nodes such that the derivative of the root with respect to these nodes, from the initial downward sweep, equals 0. This condition implies that an inactive node does not affect the expected utility of s^*, because there is 0 probability of being in that situation. Active max nodes are those max nodes in the circuit that are not inactive.

Admissible Interval Algorithm: Given a decision circuit constructed for an influence diagram and evaluated at evidence $\mathbf{e}'$, with optimal strategy s^* and maximal expected utility MEU. Determine tight and weak bounds on admissible intervals, ${I_k}^T$ and ${I_k}^W$ for all meta-parameters τ_k, $k = 1, 2, ..., K$.

1. Initialize all intervals ${I_k}^T$ and ${I_k}^W$ to $[0, 1]$.
2. Consider active max node v in the circuit. Find the partial double derivatives for all alternatives (as described in section 5) by sweeping down the subcircuit rooted at v.
3. Find intervals for all meta-parameters over which the current optimal alternative remains optimal (as described in section 5). Take the intersection of these intervals with the corresponding intervals ${I_k}^W$ from the previous iteration and reset ${I_k}^W$ to these new intervals.
4. Repeat steps 2 and 3 above for inactive max node v in the circuit, using ${I_k}^T$ in this case instead of ${I_k}^W$.
5. Repeat steps 2 and 3 above for all active and inactive max nodes in the decision circuit, updating the two corresponding kinds of intervals.
6. Set ${I_k}^T$ to be the intersection of the corresponding intervals ${I_k}^T$ and ${I_k}^W$ that were computed from previous steps.

If there are no inactive max nodes then clearly ${I_k}^W = {I_k}^T$, and this interval is the exact admissible interval. We now prove the correctness of this algorithm.

Theorem 3. *If τ_k lies in the interval ${I_k}^T$ then s^* stays optimal and if τ_k does not lie in the interval ${I_k}^W$ then s^* is no longer optimal.*

Proof. Consider any meta-parameter τ_k and its associated interval ${I_k}^T$. For $\tau_k \in {I_k}^T$, the equations in step 3 hold for all max nodes, since the resulting interval ${I_k}^T$ is the intersection of intervals. Thus s^* is the optimal strategy when $\tau_k \in {I_k}^T$. Now if $\tau_k \notin {I_k}^W$, there must be an active max node v such that the current optimal alternative for v is no longer optimal. Therefore, s^* can no longer be the optimal strategy. When $\tau_k \notin {I_k}^T$ and $\tau_k \in {I_k}^W$, we cannot be sure about whether s^* stays optimal because it is possible for an inactive node to become active for the new value of τ_k. □

Results from applying the algorithm to the example shown in Figure 2 are presented in Table 1. Two meta-parameters are considered for sensitivity analysis: the probability of the weather being sunny, $\tau_1 = \theta_w$, and the *specificity* of the report, $\tau_2 = \theta_{\bar{r}|\bar{w}}$, which is a measure of the expected number of false positives from the report. The table presents tight and weak intervals for both meta-parameters. The exact admissible interval for τ_1 is $[0.44, 0.84]$, as can be seen from Figure 4. Note that this interval contains the tight interval but lies within the weak interval. The algorithm is able to compute the exact admissible interval for τ_2 since the specificity of the report does not affect the value when evidence is not gathered.

Table 1: Results for the admissible interval algorithm for an example.

Metaparameter description	Tight interval (${I_k}^T$)	Weak interval (${I_k}^W$)
$\tau_1 = \theta_w$	$[.44, .67]$	$[.44, .89]$
$\tau_2 = \theta_{\bar{r}\|\bar{w}}$	$[.57, 1]$	$[.57, 1]$

The admissible interval algorithm yields other useful results as a by-product. If the decision maker is interested in analyzing which alternative is optimal at a particular max node in the circuit, given that all other policies are made by the current optimal strategy, then this is easy to compute with the help of the intermediate steps in the algorithm. In other words, the algorithm helps in analyzing *neighbouring strategies* to s^*, defined as strategies that differ from s^* only by an alternative in one active max node of the decision circuit. For instance, the decision maker could analyze the optimal policy at the first decision by comparing all strategies to s^* that differ in only one alternative at a max node in the decision circuit corresponding to the first decision node in the influence diagram.

The binary search method from the previous section directly applies in extensive form influence diagrams, for finding admissible intervals for all meta-parameters. It computes these intervals exactly, one meta-parameter at a time. The admissible interval algorithm identifies bounds for these intervals, but for all meta-parameters simultaneously. We suggest that the analyst use the admissible interval algorithm as an initial test to identify potentially critical meta-parameters, and then use the binary search technique to focus attention on specific meta-parameters.

6.3 Value of information

The challenge of performing value of information analysis for extensive form influence diagrams is that it is difficult to keep track of all the derivatives for all the strategies. We propose a simple method to find the value of information for variables that are not affected by any decision, if they were to be observed before the first decision is made. The goal is to re-use the decision circuit that was formulated in the offline phase.

Assuming that the decision circuit has been evaluated and therefore that the probability of evidence $P(\mathbf{e})$ has already been computed, we can find the value of information for any variable X with the help of some upward sweeps. If we pass the evidence $\mathbf{e}, U = 1, x$ in an upward sweep, once each for every possible value of x, we obtain MEU^{PI} (the maximal expected utility with perfect information on uncertain variable X), as shown in the following theorem. We state the theorem without proof; the proof is similar to Theorem 1, part (ii).

Theorem 4. *If we sweep upward with evidence* $\mathbf{e}, U = 1, x$*, for every possible value of* x*, we obtain* $MEU^{PI} = \frac{1}{P(\mathbf{e})} \sum_x g(U = 1, x, \mathbf{e})$.

If the number of variables analyzed for value of information is $\sharp var$, and assuming that the maximum number of possibilities for all of these variables is bounded by some constant, then the time complexity for obtaining the value of information for all these variables is $O((dc)(\sharp var))$. For the example in Figure 2, the value of information on the weather is: $\$ \left[u^{-1}\left(g(U = 1, w) + g(U = 1, \bar{w})\right) - 52.5\right]$, which equals \$21.3. The report already provides some information about the weather, and further information is not worth more than \$21.3.

7 CONCLUSIONS AND FUTURE RESEARCH

Decision circuits are a graphical representation for the efficient analysis of influence diagrams, with the potential to exploit both the conditional independence in the graph as well as the local structure from asymmetry of real-world decision problems. Recent methods have been devised to create compact circuits using separability of the value function and operations such as pruning and coalescence [Bhattacharjya and Shachter, 2008]. If the analyst can compile an efficient decision circuit for an influence diagram, she can then use the compiled circuit to evaluate the optimal strategy and the CE, before performing sensitivity analyses of the kind demonstrated in this paper.

We explored several one-way sensitivity analysis queries and demonstrated techniques to answer them using decision circuits. We discussed sensitivity analysis for both normal form and extensive form influence diagrams. Sensitivity analysis is an essential technique to support decision problem modeling and it provides valuable insight about the critical assessments. n-way sensitivity analysis of influence diagrams can also be performed with decision circuits using higher order derivatives, similar to the techniques discussed in this paper and in Chan and Darwiche (2004). Likewise, the presented method for computing value of information can be extended to multiple variables, by sweeping evidence corresponding to every instantiation of the variables under consideration. Decision circuits allow a wide variety of queries about the model that can be addressed efficiently by decision circuit evaluation and differentiation.

Acknowledgements

This research was partially funded by the Energy Modeling Forum. We thank John Weyant for his support, Mark Chavira for helpful answers to our queries, Adnan Darwiche for providing relevant references and the anonymous reviewers for their feedback.

References

Bhattacharjya, D., and Shachter, R., 2007, Evaluating influence diagrams with decision circuits, In Parr, R., and van der Gaag, L., editors, *Proceedings of the Twenty-third Conference on Uncertainty in Artificial Intelligence*, pp. 9–16, Vancouver, BC, Canada: AUAI Press.

Bhattacharjya, D., and Shachter, R., 2008, Dynamic programming in decision circuits, Working paper.

Castillo, E., Gutierrez, J. M., and Hadi, A. S., 1997, Sensitivity analysis in discrete Bayesian networks, *IEEE Transactions on Systems, Man, and Cybernetics*, **27**, pp. 412-423.

Chan, H., and Darwiche, A., 2002, When do numbers really matter?, *Journal of Artificial Intelligence Research*, **17**, pp. 265–287.

Chan, H., and Darwiche, A., 2004, Sensitivity analysis in Bayesian networks: From single to multiple parameters, In Chickering, M., and Halpern, J., editors, *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, pp. 67–75, Banff, Canada: AUAI Press, Arlington, Virginia.

Chan, H., and Darwiche, A., 2006, On the robustness of Most Probable Explanations, In *Proceedings of the Twenty Second Conference on Uncertainty in Artificial Intelligence*, pp. 63–71, Cambridge, MA, USA: Morgan Kaufmann, San Mateo, California.

Chavira, M., and Darwiche, A., 2005, Compiling Bayesian networks with local structure, In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, pp. 1306–1312.

Darwiche, A., 2000, A differential approach to inference in Bayesian networks, In Boutilier, C., and Goldszmidt, M., editors, *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pp. 123–132, Stanford, CA, USA: Morgan Kaufmann, San Mateo, California.

Darwiche, A., 2002, A logical approach to factoring belief networks, In *Proceedings of International Conference on Knowledge Representation and Reasoning*, pp. 409–420.

Darwiche, A., 2003, A differential approach to inference in Bayesian networks, *Journal of the ACM*, **50(3)**, pp. 280–305.

Howard, R., 1966, Information value theory, *IEEE transactions on Systems Science and Cybernetics*, **2**, pp. 22–26.

Howard, R., 1983, The evolution of decision analysis, In Howard, R., and Matheson, J., editors, 1989, *The Principles and Applications of Decision Analysis*, Strategic Decisions Group, Menlo Park, CA.

Howard, R., and Matheson, J., 1984, Influence diagrams, In Howard, R., and Matheson, J., editors, *Influence diagrams, belief nets, and decision analysis*, pp. 3–23, Wiley, Chichester.

Kjaerulff, U., van der Gaag, L. C., 2000, Making sensitivity analysis computationally efficient, In Boutilier, C., and Goldszmidt, M., editors, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pp. 317-325, Stanford, California: Morgan Kaufmann, San Francisco, California.

Laskey, K. B., 1995, Sensitivity analysis for probability assessments in Bayesian networks, *IEEE Transactions on Systems, Man, and Cybernetics*, **25**, pp. 901-909.

Nielsen, T., and Jensen, F. V., 2003, Sensitivity analysis in influence diagrams, *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, **33** No. 1, March, pp. 223–234.

Raiffa, H., 1968, *Decision Analysis: Introductory Lectures on Choices under Uncertainty*, Addison-Wesley.

Savage, L., 1954, *The Foundations of Statistics*, Wiley, New York.

Shachter, R., 1986, Evaluating influence diagrams, *Operations Research*, **34** (November-December), pp. 871–882.

Shachter, R., 2007, Model building with belief networks and influence diagrams, In Edwards, W., Miles, R., and von Winterfeldt, D., editors, *Advances in Decision Analysis: From Foundations to Applications*, pp. 177-201, Cambridge University Press.

van der Gaag, L. C., and Renooij, S., 2001, Analysing sensitivity data from probabilistic networks, In Breese, J., and Koller, D., editors, *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pp. 530-537, Seattle, WA: Morgan Kaufmann, San Francisco, California.

von Neumann, J., and Morgenstern, O., 1947, *Theory of Games and Economic Behavior*, 2nd edition, Princeton University Press, Princeton, NJ.

Greedy Block Coordinate Descent for Large Scale Gaussian Process Regression

Liefeng Bo[1]
[1]Toyota Technological Institute at Chicago
Chicago, IL 60637
blf0218@tti-c.org

Cristian Sminchisescu[2,1]
[2]University of Bonn
Bonn, 53115, Germany
sminchisescu.ins.uni-bonn.de

Abstract

We propose a variable decomposition algorithm–greedy block coordinate descent (GBCD)–in order to make *dense* Gaussian process regression practical for large scale problems. GBCD breaks a large scale optimization into a series of small sub-problems. The challenge in variable decomposition algorithms is the identification of a sub-problem (the active set of variables) that yields the largest improvement. We analyze the limitations of existing methods and cast the active set selection into a zero-norm constrained optimization problem that we solve using greedy methods. By directly estimating the decrease in the objective function, we obtain not only efficient approximate solutions for GBCD, but we are also able to demonstrate that the method is globally convergent. Empirical comparisons against competing dense methods like Conjugate Gradient or SMO show that GBCD is an order of magnitude faster. Comparisons against sparse GP methods show that GBCD is both accurate and capable of handling datasets of 100,000 samples or more.

1 Introduction

Solving linear systems is frequently encountered in least squares kernel methods. A relevant example is Gaussian process regression (GPR) with Gaussian noise (Williams & Rasmussen, 1996; Rasmussen & Williams, 2006), a method that has become increasingly popular in the field of machine learning. Given a set of training samples $\{\mathbf{x}_i\}_{i=1}^n$ along with the corresponding targets $\{y_i\}_{i=1}^n$, the predictive mean and variance of the estimator can be computed in closed form

$$m(\mathbf{x}_*) = \mathbf{k}_*^\top \left(\mathbf{K} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y} \tag{1}$$

$$v(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) + \sigma^2 - \mathbf{k}_*^\top \left(\mathbf{K} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{k}_* \tag{2}$$

where $k(\mathbf{x}_i, \mathbf{x}_j)$ is a positive definite kernel function, $\mathbf{K}$ is a $n \times n$ matrix with $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{k}_*$ is a $n \times 1$ vector with the i-th component being $\mathbf{k}_* = [k(\mathbf{x}_*, \mathbf{x}_i)]$, and σ^2 is noise variance. GPR requires an $n \times n$ matrix inversion with $O(n^3)$ training cost and $O(n^2)$ requirements for memory storage, which is non-trivial since the kernel matrix $\mathbf{K}$ can not be fitted into memory for large datasets (this is known as the out-of-memory case). Many standard linear system solvers, such as Cholesky factorization, implicitly assume the storage of $\mathbf{K}$ is possible, which limits their applicability in this case. Alternatively, one can recompute $\mathbf{K}$ on the fly, but this becomes computationally prohibitive if $\mathbf{K}$ is frequently needed.

Instead of directly inverting the covariance matrix, one can use iterative methods such as the conjugate gradient algorithm (CG) in order to train GPR. This starts with an initial guess and modifies the current solution iteratively until a given stopping condition is satisfied. CG has speed of convergence guarantees in that it takes no more than n steps to reach the exact solution. In practice, the actual number of iterations necessary for a given precision is often much smaller than n, hence CG is usually faster than direct matrix inversion. Unfortunately, CG is again not well adapted for GPR in the out-of-memory case because expensive covariance matrix re-evaluations are necessary inbetween iterations.

Another way to attack the out-of-memory case is the block coordinate descent (BCD) method (Bertsekas, 1999), known as the block Gauss-Seidel for linear systems (Ortega & Rheinboldt, 1970). At each iteration, block coordinate descent splits variables into two subsets, the set of the active variables and the set of inactive ones, then minimizes the objective function along active dimensions while inactive variables are fixed at current values. Block coordinate descent is attractive since only the kernel matrix indexed by the active set needs to be (re)evaluated, thus significantly reducing the cost per iteration.

The efficiency of block coordinate descent methods depends strongly on the active set selection. A frequently used method is to select the active set in cyclical order.

Beatson et al. (2000) applied the cyclical block coordinate descent for solving radial basis function interpolation equations and Li et al. (2007) adopted it to train regularized least squares classifiers. The potential weakness of cyclical block coordinate descent is that it does not exploit any information about the objective function in the process of selecting the active set. The gradient-based active set selection is an alternative method that maximizes the gradient norm (Zoutendijk, 1970; Joachims, 1999). However, the gradient-based active set selection still does not directly relate the variable selection and the decrease in the objective function, leading to non-optimal updates.

In this paper, we propose a greedy block coordinate descent (GBCD) method in order to improve the selection of sub-problems solved during optimization. We show that active set selection can be cast as a zero-norm constrained optimization problem. While the exact solution requires combinatorial search, we show that greedy algorithms can be used to obtain approximate solutions with $O(m^3)$ cost, where m is the size of sub-problems. This maintains the same computational complexity as gradient-based active set selection, per iteration, yet it achieves significant speedups. Differently from existing active set methods that select all active variables simultaneously, our method constructs the active set incrementally. Hence, it can build upon the current optimization context in order to implicitly avoid redundancy in previously selective active variables. This avoids inefficiencies due to the inclusion of high correlated variables in the active set. Our experiments show that GBCD compares favorably with existing methods, both dense and sparse (and against the constraints expected in each case, i.e. accuracy vs. speed). We do not only present experiments on standard machine learning benchmarks but also on real-world large scale computer vision applications like the reconstruction of three-dimensional human motions like walking, running, gestures or boxing from image sequences acquired with a single video camera, as available in the HumanEva dataset (Sigal & Black, 2006).

1.1 Related Work

Sequential minimal optimization: Decomposition algorithms (Platt, 1999; Keerthi et al., 2001) have been widely used in the field of support vector machines (SVMs). Sequential minimal optimization (SMO) (Platt, 1999) is the most prominent method in which the working set only includes two variables, hence the sub-problems can be solved analytically. The most popular way to select the active set for SMO is the maximal violating pair method, first proposed by (Keerthi et al., 2001) and used in many SVMs packages such as LIBSVM 2.71 (Chang & Lin, 2001). Keerthi and Shevade (2003) extended SMO to least square support vector machines, a problem very similar with the one we consider here, hence SMO also can be adopted for GPR. SMO differs from our method in the way the active variables are chosen and in the size of sub-problems.

Fast multipole methods: The most time-consuming step in CG is the multiplication of a kernel matrix with a vector. This can be sped up by fast multipole methods and KD-Trees. In particular, Yang et al. (2005) and Shen et al. (2006) have applied this method to Gaussian process regression. While the methods appear to be effective on low-dimensional problems, they have not been demonstrated yet for high dimensional problems. The quality of Hermite or Taylor approximations used may also degrade exponentially as a function of the dataset dimensionality.

Sparse GPR models: Recently, there has been substantial research on deriving sparse approximations to the full GPR (Smola & Bartlett, 2001; Csató & Opper, 2002; Lawrence et al., 2003; Seeger et al., 2003; Keerthi et al., 2006; Keerthi & Chu, 2006) that reduce the training and storage cost. The methods select a representative subset of regressors, thus dropping the training complexity to $O(np^2)$, where p is the size of the subset. Since $p \ll n$ in most cases, sparse approximations achieve substantial speedups relative to the full GPR. Though very appealing, sparse approximations are not always the best choice in applications that require high accuracy. Indeed, recent research (Rasmussen & Candela, 2005; Quiñonero-Candela & Rasmussen, 2005) confirms that sparse GP can sometimes lead to unreasonable predictive distributions. Our experiments also show that sparse GP methods can sometimes be significant less accurate compared to the full GPR model.

2 Greedy Block Coordinate Descent

It is well-known that the solution to the positive definite linear system (1) and (2) can be obtained from a quadratic optimization problem

$$\min_{\boldsymbol{\alpha}} \left[f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^\top \bar{\mathbf{K}}\boldsymbol{\alpha} - \mathbf{y}^\top \boldsymbol{\alpha} \right] \tag{3}$$

where $\bar{\mathbf{K}} = \mathbf{K} + \sigma^2\mathbf{I}$.

Let $\mathbf{g}^k = \bar{\mathbf{K}}\alpha^k - \mathbf{y}$ be the current gradient vector. A gradient-based method selects the active set B by solving the following optimization problem

$$\max_{|B|=m, B\subseteq\{1,\cdots,n\}} |\mathbf{g}^k_B| \tag{4}$$

where m is the size of active set. A key observation is that the gradient-based active set selection is not directly related with the decrease in the objective function, possibly leading to non-optimal improvement. In contrast, we integrate the active set selection into the solution to sub-problems. At each iteration of decomposition, it is desirable to decrease the objective function as much as possible for a given size of the active set. The zero-norm formulation of this prob-

Algorithm 1 Greedy Block Coordinate Descent

1. Set $k = 0$, $\boldsymbol{\alpha}^k = \mathbf{0}$, and $\mathbf{g}^k = -\mathbf{y}$
2. If $\boldsymbol{\alpha}^k$ is an optimal solution of (3), stop; otherwise go to step 3
3. Solve (6) using greedy algorithm and obtain $\Delta\boldsymbol{\alpha}^{opt}$ and the active set B
4. Set $\boldsymbol{\alpha}_B^{k+1} = \boldsymbol{\alpha}_B^k + \Delta\boldsymbol{\alpha}_B^{opt}$, $\boldsymbol{\alpha}_N^{k+1} = \boldsymbol{\alpha}_N^k$, and $\mathbf{g}^{k+1} = \mathbf{g}^k + \mathbf{K}_B\Delta\boldsymbol{\alpha}_B^{opt}$ where $\mathbf{K}_B$ is the sub-matrix made of the columns indexed by B and $\boldsymbol{\alpha}_B^k$ is the sub-vector indexed by B. Let $k = k + 1$ and go back to step 2

lem can be written as

$$\min_{\Delta\boldsymbol{\alpha}} \left[\frac{1}{2}(\boldsymbol{\alpha}^k + \Delta\boldsymbol{\alpha})^\top \bar{\mathbf{K}}(\boldsymbol{\alpha}^k + \Delta\boldsymbol{\alpha}) - \mathbf{y}^\top(\boldsymbol{\alpha}^k + \Delta\boldsymbol{\alpha})\right] \quad s.t. \quad \|\Delta\boldsymbol{\alpha}\|_0 = m \tag{5}$$

where $\|\cdot\|_0$ is the zero-norm, counting the nonzero entries of a vector, and m is the number of nonzero entries, i.e. the size of the active set. Eliminating the constant term in (5), we obtain

$$\min_{\Delta\boldsymbol{\alpha}} \left[\frac{1}{2}\Delta\boldsymbol{\alpha}^\top \bar{\mathbf{K}}\Delta\boldsymbol{\alpha} + (\mathbf{g}^k)^\top\Delta\boldsymbol{\alpha}\right] \quad s.t. \quad \|\Delta\boldsymbol{\alpha}\|_0 = m \tag{6}$$

where $\mathbf{g}^k = \mathbf{g}(\boldsymbol{\alpha}^k) = \bar{\mathbf{K}}\boldsymbol{\alpha}^k - \mathbf{y}$. There are several difficulties in solving (6). First, the constraint is not differentiable, so gradient descent algorithms can not be used. Second, the optimizers can get trapped in a shallow local minimum because the cost (6) is not necessary convex. An exhaustive search over all possible choices ($\|\Delta\boldsymbol{\alpha}\|_0 = m$) is expensive as the number of possible combinations $\binom{n}{m}$ is usually too large to enumerate even on current computers. One option is to use more sophisticated search algorithms such as branch-and-bound in order to decrease the cost of exhaustive search. But, branch-and-bound is still too expensive for large n, and it seems unwise to spend too much work on sub-problems anyway. In fact, although the approach can decrease the objective faster and may take fewer iterations to converge, the overall training time may not be reduced since each iteration is expensive.

A more practical view is to decrease the objective function as much as possible with little extra work. In this paper, we compute an approximate solution to (6) using a greedy algorithm. This gives a balanced tradeoff between the decrease in objective function and the computational cost of each iteration, and converges fast in our experiments. We refer to this novel decomposition algorithm as Greedy Block Coordinate Descent (GBCD), and its steps are described in the table associated with **Algorithm 1** (Greedy Block Coordinate Descent).

2.1 Greedy Approximation

Unlike cyclical and gradient-based active set methods that select all variables simultaneously, our greedy algorithm selects the active variables incrementally. Starting with an empty active set, our greedy optimizer selects one active variable at a time, so that the objective function is decreased. The selection process stops when the number of active variables reaches a predefined value m.

Let $B = \emptyset$ and $N = \{1, \cdots, n\}$. How do we select an active variable from N? A natural idea is to optimize the objective with respect to $\Delta\boldsymbol{\alpha}_B$ and $\Delta\alpha_i$ for each $i \in N$ and select the variable giving the largest decrease. This process leads to a two-layer optimization problem

$$s = \arg\min_{i\in N}\left[\min_{\Delta\boldsymbol{\alpha}_B,\Delta\alpha_i} \frac{1}{2}\begin{bmatrix}\Delta\boldsymbol{\alpha}_B\\ \Delta\alpha_i\end{bmatrix}^\top\begin{bmatrix}\bar{\mathbf{K}}_{BB} & \bar{\mathbf{K}}_{Bi}\\ \bar{\mathbf{K}}_{iB} & \bar{\mathbf{K}}_{ii}\end{bmatrix}\begin{bmatrix}\Delta\boldsymbol{\alpha}_B\\ \Delta\alpha_i\end{bmatrix} + \begin{bmatrix}\mathbf{g}_B^k\\ g_i^k\end{bmatrix}^\top\begin{bmatrix}\Delta\boldsymbol{\alpha}_B\\ \Delta\alpha_i\end{bmatrix}\right] \tag{7}$$

where $\bar{\mathbf{K}}_{Bi}$ is the sub-matrix made of rows indexed by B and the column indexed by i. This active set selection method, called prefitting, has appeared in the (prefitting) version of kernel matching pursuit (KMP) (Vincent & Bengio, 2002) and the sparse greedy Gaussian process (Smola & Bartlett, 2001). However, prefitting needs to solve m+1 dimensional optimizations $|N|$ times, which obviously has a higher cost than optimizing the sub-problem.

A cheaper method is to fix $\Delta\boldsymbol{\alpha}_B^t$ and optimize (7) only with respect to $\Delta\alpha_i$. This can be expressed as a two-layer optimization problem

$$s = \arg\min_{i\in N}\left[h_i = \min_{\Delta\alpha_i} \frac{1}{2}\begin{bmatrix}\Delta\boldsymbol{\alpha}_B^t\\ \Delta\alpha_i\end{bmatrix}^\top\begin{bmatrix}\bar{\mathbf{K}}_{BB} & \bar{\mathbf{K}}_{Bj}\\ \bar{\mathbf{K}}_{iB} & \bar{\mathbf{K}}_{ii}\end{bmatrix}\begin{bmatrix}\Delta\boldsymbol{\alpha}_B^t\\ \Delta\alpha_i\end{bmatrix} + \begin{bmatrix}\mathbf{g}_B^k\\ g_i^k\end{bmatrix}^\top\begin{bmatrix}\Delta\boldsymbol{\alpha}_B^t\\ \Delta\alpha_i\end{bmatrix}\right] \tag{8}$$

Eliminating the constant in (8), we get

$$h_i = \min_{\Delta\alpha_i}\left[\frac{1}{2}\left(k(\mathbf{x}_i,\mathbf{x}_i) + \sigma^2\right)\Delta\alpha_i^2 + e_i^t\Delta\alpha_i\right], \quad i \in N \tag{9}$$

where $e_i^t = \bar{\mathbf{K}}_{iB}\Delta\boldsymbol{\alpha}_B^t + g_i^k$. This is a one dimensional quadratic programming problem and can be solved analytically. Thus, we can determine the new active variable by the formula

$$s = \arg\min_{i\in N}\left[h_i = \frac{-(e_i^t)^2}{2(k(\mathbf{x}_i,\mathbf{x}_i) + \sigma^2)}\right] \tag{10}$$

In solving the sub-problems, the inversion of the covariance matrix is the computational bottleneck. In the $(t+1)$-th iteration, the inversion of the covariance indexed by the current active set can be written as

$$\mathbf{R}^{t+1} = \begin{bmatrix}\bar{\mathbf{K}}_{BB} & \bar{\mathbf{k}}_s\\ \bar{\mathbf{k}}_s^\top & \bar{\mathbf{K}}_{ss}\end{bmatrix}^{-1} \tag{11}$$

where $\bar{\mathbf{k}}_s = [\bar{\mathbf{K}}_{b_1 s}, \bar{\mathbf{K}}_{b_2 s}, \cdots, \bar{\mathbf{K}}_{b_i s}]$. Applying the Woodbury inversion identity (Stoer & Bulirsch, 1993) to (11), we get

Algorithm 2 Greedy Approximation with Random Subset

1. Set $t=0$, $\Delta\boldsymbol{\alpha}^t=\mathbf{0}$, $\mathbf{e}^t=\mathbf{g}^k$, $B=\emptyset$, and $O=N=\{1,\cdots,n\}$
2. If $t=m$, stop; otherwise go to step 3
3. $s=\arg\min_{i\in O}\left[\frac{-(e_i^t)^2}{2(k(\mathbf{x}_i,\mathbf{x}_i)+\sigma^2)}\right]$
4. If t=0, $\mathbf{R}^{t+1}=\frac{1}{k(\mathbf{x}_i,\mathbf{x}_i)+\sigma^2}$ and $\Delta\alpha_s=\frac{-e_s}{k(\mathbf{x}_i,\mathbf{x}_i)+\sigma^2}$, otherwise compute $\mathbf{R}^{t+1}$, $\Delta\boldsymbol{\alpha}_B^{t+1}$, and $\Delta\alpha_s^{t+1}$ according to (12) and (13)
5. Set $B=B+\{s\}$, $N=N-\{s\}$
6. Randomly choose a subset O of size κ from N. Let $\mathbf{e}_O^{t+1}=\bar{\mathbf{K}}_{OB}\Delta\boldsymbol{\alpha}_B^{t+1}+\mathbf{g}^k$, $t=t+1$, and go to step 2

an update formula

$$\mathbf{R}^{t+1}=\begin{bmatrix}\mathbf{R}^t & \mathbf{0}\\ \mathbf{0}^\top & 0\end{bmatrix}+\eta\begin{bmatrix}\boldsymbol{\beta}\\ -1\end{bmatrix}\begin{bmatrix}\boldsymbol{\beta}^\top & -1\end{bmatrix} \tag{12}$$

where $\boldsymbol{\beta}=\mathbf{R}^t\bar{\mathbf{k}}_s$, $\eta=(\bar{\mathbf{K}}_{ss}-\bar{\mathbf{k}}_s^\top\boldsymbol{\beta})^{-1}$. Combining (12) and $\Delta\boldsymbol{\alpha}_B^t=-\mathbf{R}^t\mathbf{g}_B^k$, we obtain the update formula for $\Delta\boldsymbol{\alpha}^{t+1}$

$$\begin{bmatrix}\Delta\boldsymbol{\alpha}_B^{t+1}\\ \Delta\alpha_s^{t+1}\end{bmatrix}=-\mathbf{R}^{t+1}\begin{bmatrix}\mathbf{g}_B^k\\ g_s^k\end{bmatrix}=\begin{bmatrix}\Delta\boldsymbol{\alpha}_B^t\\ 0\end{bmatrix}-\eta\left(\boldsymbol{\beta}^\top\mathbf{g}_B^k-g_s^k\right)\begin{bmatrix}\boldsymbol{\beta}\\ -1\end{bmatrix} \tag{13}$$

(12) and (13) indicate that we can efficiently update $\mathbf{R}^{t+1}$ and $\Delta\boldsymbol{\alpha}^{t+1}$ at a cost of $O(t^2)$ without explicitly computing the inverse matrix. The update formula (12) is numerical stable since the regularization term $\sigma^2\mathbf{I}$ greatly improves the condition number of the matrix $\mathbf{K}$. One can improve the numerical stability further, by using Cholesky decomposition (Stoer & Bulirsch, 1993). Integrating the active set selection and the solution of the sub-problems, we obtain the greedy approximation algorithm shown in the table associated with **Algorithm 2** (Greedy Approximation with Random Subset).

The greedy approximation involves three operations: 1) computing the new column of the covariance which is $O(cn)$, where c is the cost of evaluating the kernel function one time, 2) updating $\mathbf{e}^t$ which is $O(nm)$, and 3) selecting the new active variable according to (10) which is $O(n)$. The cost of solving sub-problems is dominated by updating the inverse of the covariance sub-matrix which is $O(m^2)$. Adding up, the cost of choosing m active variables gives the overall complexity $O(cnm+nm^2+m^3)$.

For large m values, updating $\mathbf{e}^t$ dominates the cost of greedy optimizer, which is still more than what we want to accept. This cost can be reduced to $O(\kappa m^2)$ by only considering the random subset O of N with size κ and selecting the active variables only from O rather than exhaustively searching the full set N. Note that the number of covariance function evaluations at each iteration is still $O(cnm)$ because step 4 of GBCD requires computing a covariance matrix indexed by the active set.

2.2 Convergence

Theorem 1: Let

$$\lambda_{min}=\min_{B\subseteq\{1,\cdots,n\}:|B|=m}\left[\min eig\left(\bar{\mathbf{K}}_{BB}\right)\right] \tag{14}$$

where $\min[eig(\bar{\mathbf{K}}_{BB})]$ denotes the smallest eigenvalue of the matrix $\bar{\mathbf{K}}_{BB}$. The following inequation holds

$$f\left(\boldsymbol{\alpha}^{k+1}\right)-f\left(\boldsymbol{\alpha}^k\right)\leq-\frac{1}{2}\lambda_{min}\left\|\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^k\right\|_2^2 \tag{15}$$

Theorem 1 indicates that a random chosen active set can give the objective function (3) sufficient decrease if the covariance function is positive definite.

Theorem 2: $\{\boldsymbol{\alpha}^k\}$ converges to the global minimum of (3).

Proofs to Theorems 1 and 2 are given in the Appendix.

2.3 Intuitions on Active Set Selection

This section analyses the efficiency of decomposition algorithms and motivates why the greedy active set selection we propose works better than competing methods. We attribute the efficiency of our decomposition to its gains over *gradient correlation*: if inputs $\mathbf{x}_i$ and $\mathbf{x}_j$ are neighbors, their gradients g_i and g_i are correlated with high probability. In contrast, our method selects variables with *uncorrelated* gradients. We validate this conjecture further by means of a quantitative experiment, where we randomly choose an input $\mathbf{x}_i$ and find its 50 neighbors based on the Euclidean distance between $\mathbf{x}_i$ and the other inputs. We then track the optimization process of GBCD and record the gradient of $\mathbf{x}_i$ and its neighbors across 50 successive iterations. Finally, we compute correlation coefficients for the gradient of $\mathbf{x}_i$ and its neighbors using $\frac{\text{cov}(g_i,g_j)}{\text{std}(g_i)\text{std}(g_j)}$. We repeat the process over 100 inputs randomly chosen from training set. Counting the frequency of correlation coefficients falling in different intervals, we obtain the histograms in fig. 1. One can see that the gradients of $\mathbf{x}_i$ and its neighbors are non-negligibly correlated.

Intuitively, one should select the active set in a way that makes the gradients as uncorrelated as possible, because this brings in diversity from variables outside the active set. This explains our experimental findings: the cyclical active set selection works better than the gradient-based one. The latter works by selecting variables that maximize the gradient infinite norm, thus it tends to select many correlated variables, which prevents rapid progress during optimization. The cyclical method avoids this on average, because it is random; similarly does SMO, which selects the maximal violating pair as the active set—this is usually uncorrelated with high probability. Our greedy method selects the active set incrementally, hence it exploits information in the

Table 1: Number of training and test samples, and size of attribute vector from benchmark datasets. No matter the algorithm, the samples we test on are never used for validation or training.

Problem	Training	Test	Attribute
Calhouse	18000	2640	8
Outaouais	20000	9000	37
Kin40k	30000	10000	8
Sarcos	44484	4449	21
Friedman1	100000	5000	10

previously chosen active variables and avoids the selection of highly correlated ones. The gradients corresponding to variables that are highly correlated with ones previously selected are usually small–in contrast the other greedy methods will likely select variables with simply large gradients, see (10).

3 Experiments

In this section, we empirically study the behavior of GBCD on several benchmark datasets and compare it to existing methods, both dense and sparse. The benchmarks are presented in table 1. Kin40k are available from Torgo's homepage[1], Outaouais is from the *Evaluating Predictive Uncertainty Challenge*[2], Friedman1 is from the author, Friedman (1991), Sarcos from the book *Gaussian processes for machine learning*[3]. Gaussian noise with unit standard deviation is added to the training samples. For all datasets, each attribute of the training inputs and outputs are linearly scaled to zero mean and unit variance and the same transformation is used for the test set. Unless otherwise specified, we report the normalized root mean squared error (RMSE) on the test set given by

$$RMSE = \sqrt{\frac{1}{t}\sum_{i=1}^{t}\frac{(y_i^{'} - m_i)^2}{var(\mathbf{y})}} \tag{16}$$

where $y_i^{'}$ is the output of test samples, m_i is the predictive mean and $var(\mathbf{y})$ is the variance of training samples. For algorithms that require random numbers, RMSE is averaged over 10 trials.

All algorithms are implemented in VC++ 6.0 and run on a PC with 3.6 GHz P4 processors, 2 GB memory, and Windows XP. Greedy approximations based on random search are used. The size of the random subset κ is set to 60. (Smola and Schölkopf (2000) have also found that the random set of size 59 achieved good performance). Optimization is assumed converged when the infinite norm of gradient is within 10^{-4} tolerance. The size of sub-problems is set to 500 for BCDC, BCDG and GBCD.

[1] http://www.liacc.up.pt/~ltorgo/Regression/DataSets.html
[2] http://predict.kyb.tuebingen.mpg.de
[3] http://www.gaussianprocess.org/gpml/data/

The squared exponential function $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\sum_{l=1}^{d}\gamma_t(\mathbf{x}_i^l - \mathbf{x}_j^l)^2)$ is used to construct the kernel matrix. The optimal values for hyperparameters $(\gamma_1, \cdots, \gamma_d, \sigma^2)$, are determined, for each model, by maximizing the marginal likelihood on a tractable subset of 2000 samples randomly sampled from the training set. BFGS is used to optimize the hyperparameters. The termination tolerance on the marginal likelihood and on the hyperparameters is set to 10^{-4}.

To compute the predictive mean of GPR, we only need to solve one linear system whereas in order to compute the predictive variance, we need to solve a linear system per sample. In the sequel, *training time* is measured as time required to solve the linear system $\mathbf{K} + \sigma^2\mathbf{I} = \mathbf{y}$; *mean time* is the time necessary to compute the predictive mean; *variance time* is the time needed to compute the predictive variance, which requires solving one linear system $\mathbf{K} + \sigma^2\mathbf{I} = \mathbf{k}_*$. If the computation variance is required, this will dominate test time.

3.1 Comparisons with Other Iterative Methods

This section compares CG, SMO, Cholesky factorization, BCDC, BCDG and GBCD. BCDC selects an active set using the cyclic order and BCDG selects an active set using gradient information. We only use a subset of 10,000 samples randomly chosen from training samples for all algorithms, which allows us store the full covariance matrix and solve GPR using Cholesky factorization. Cholesky factorization serves as baseline, in order to know whether the solutions obtained by the iterative algorithms we test are close to the exact one. We do not include the gradient-based active set selection method because the cyclic method outperforms it in all of our experiments. For CG, SMO, BCDC, BCDG and GBCD, we re-calculate the covariance matrix whenever needed in order to simulate the out-of-memory case.

Table 2 gives the training time and RMSE of four algorithms. GBCD is significantly faster than CG, SMO, BCDC and BCDG on all the datasets. BCDG is significantly slower than other methods because of selection of many correlated variables. In particular, for Sarcos, GBCD is 10 times faster than BCDC, 20 times faster than SMO and 59 times faster than CG. CG, SMO, BCDC, BCDG and GBCD have RMSE accuracy that is similar to the result of Cholesky factorization up to 3 significant digits. This confirms that our iterative algorithms find good solutions for the convergence tolerances used.

Fig. 2 shows the RMSE, gradient infinite norm and objective function on the test set, for all algorithms as function of training time. We do not include the test error of CG and BCDG because these give significantly higher test errors at the initial stages of optimization and affects the scaling / readability of plots even on log-scale. GBCD con-

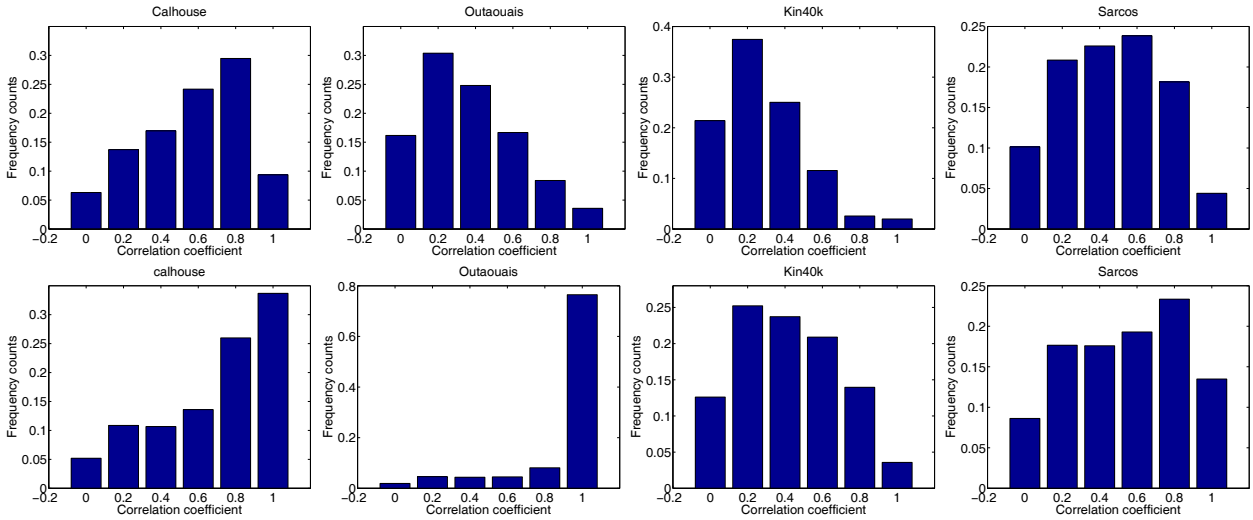


Figure 1: Frequency of correlation coefficients of neighboring points, histogramed. Plots on top row are obtained by solving the linear system $\mathbf{K}+\sigma^2\mathbf{I}=\mathbf{y}$, the ones on the bottom row are obtained by solving the linear system $\mathbf{K}+\sigma^2\mathbf{I}=\mathbf{k}_*$. Correlation coefficients for the regime shown on the bottom are higher than the corresponding ones on the top. This indicates that the linear system $\mathbf{K}+\sigma^2\mathbf{I}=\mathbf{k}_*$ is easier to solve, which is consistent with our experiments, discussed in §3.

verges significantly faster than other algorithms in terms of both the objective function decrease and the gradient infinite norm. GBCD usually achieves stable test accuracy far before the specified convergence criterion is reached. This indicates that a looser criterion can be used when training time constraints exist.

Fig. 3 plots the gradient infinite norm and the objective function across iterations. As expected, greedy active set selection decreases the objective significantly faster than active set section based on cyclic ordering, or gradient based active set selection, further confirming our analysis given in §2. 2.

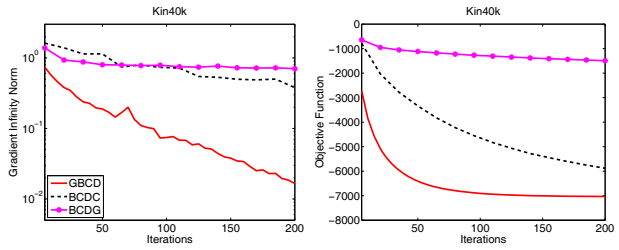


Figure 3: Objective function and infinite gradient norm among iterations. Notice that GBCD gives significantly faster objective decrease than BCDC and BCDG.

Fig. 4 plots training time as function of the size of dataset Friedman1. Notice that the gap between BCDC and GBCD further increases with the size of the dataset, showing that it is impractical to train large scale models using BCDC.

Table 3 gives the averaged variance time of GBCD on each test sample and RMSE of the predictive variance relative to

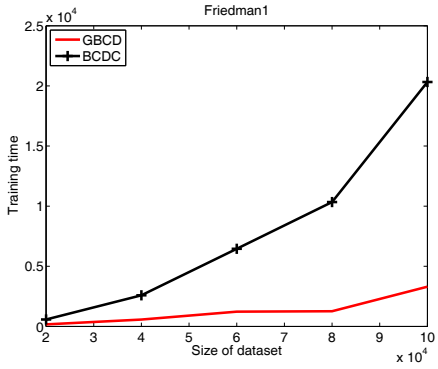


Figure 4: Training time as a function of the dataset size for different models. The training time gap widens as the dataset increases, showing that it is impractical to train BCDC models in this case. BCDC and GBCD are assumed converged when the infinite gradient norm is smaller than 10^{-4}.

Cholesky factorization

$$\sqrt{\frac{1}{t}\sum_{i=1}^{t}\left(\frac{v_i'-v_i}{v_i'}\right)^2} \tag{17}$$

where v_i' is the predictive variance of Cholesky, v_i is the predictive variance of GBCD. As one can see, the relative error is 0.02 in the worst case. This indicates that the predictive variance of GBCD is very close to that of Cholesky factorization. We are not able to report the relative RMSE of the predictive variance of the other algorithms relative to that of Cholesky, due to extremely long runtime. Unlike the predictive mean where one only needs to solve a linear system, one needs to solve different linear systems for each

Table 2: Training time and RMSE on test set of CG, SMO, BCDC, BCDG and GBCD. Chol denotes Cholesky factorization. Optimization is stopped when the infinite norm of the gradient is smaller than 10^{-4}. '/' show that the values are not available. Bold indicates the lowest training time among all iterative methods. Cholesky factorization stores the full covariance matrix while other method don't.

	Training Time (second)						Root Mean Squared Error					
	Chol	CG	SMO	BCDC	BCDG	GBCD	Chol	CG	SMO	BCDC	BCDG	GBCD
Calhouse	128	1958	231	331	25563	**92**	0.477	0.477	0.477	0.477	0.477	0.477
Outaouais	136	24408	10320	5366	>10 hours	**1154**	0.216	0.216	0.216	0.216	/	0.216
Kin40k	130	14956	5910	3677	>10 hours	**975**	0.098	0.098	0.098	0.098	/	0.098
Sarcos	132	12880	4295	2172	>10 hours	**217**	0.128	0.128	0.128	0.128	/	0.128
Friedman1	126	4732	32899	122	>10 hours	**106**	0.017	0.017	0.017	0.017	/	0.017

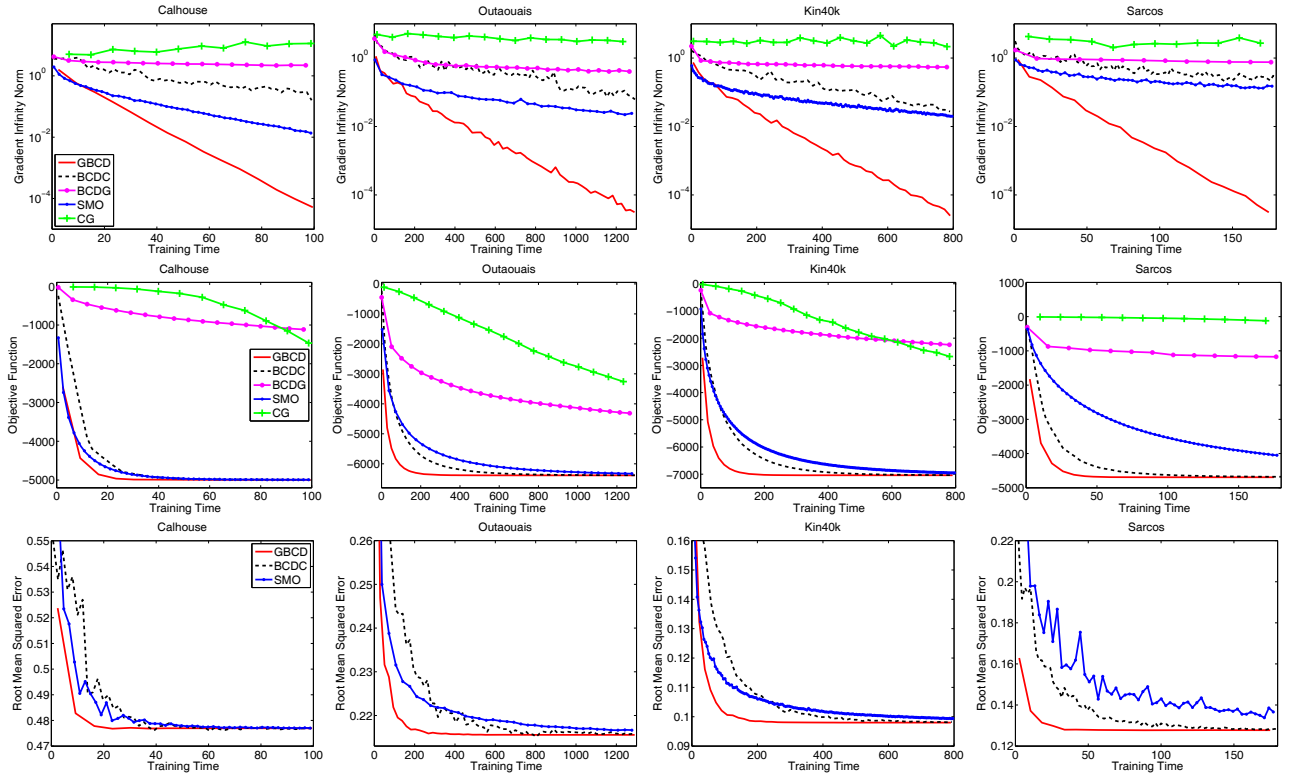


Figure 2: Infinite norm of the gradient, objective function and test error of CG, SMO, BCDC, BCDG and GBCD function of training time, in seconds. The infinite norm of the GBCD gradient converges to 10^{-4} precision earlier than competing methods. A similar behavior is observed for the objective function and the test error of GBCD, which stabilize more rapidly than for the other algorithms we have tested.

test sample in order to obtain the predictive variance.

3.2 Comparisons with Sparse GPR

This section compares GBCD and three sparse GPR models. The first is referred as SD (Subset of the Data), which estimates a GPR predictor based on a subset of size p in the training set. The second is known as SR which only uses a random subset of regressors. The third is MPGP which selects the subset of regressors using matching pursuit. The memory requirement of SR and MPGP is $O(pn)$.The size of the subset, p is set to 2000 for all five datasets.

From Table 4, we see that GBCD always achieves lower RSME than sparse GPR models. For Outaouais, GBCD achieves a 55% improvement relative to the best sparse GPR model. This is not surprising because GBCD computes the predictive mean and variance of the full GPR while sparse GPR only gives an approximation. An obvious limitation of sparse GPR models is that computationally feasible p values my lead to less accurate solutions compared to the full GPR. On the other hand, sparse GPR models are faster in testing than GBCD, hence GBCD is well fitted for for problems that require accuracy whereas sparse GPR models are suitable for applications that re-

Table 4: Training time, variance time and root mean squared error on the test set for GBCD and sparse GPR models. Time is given in seconds. All methods use the same hyperparameters, see section 3.1 for details. The *variance time* is needed to compute the predictive variance. The sparse GPR model is stopped when the specified number of regressors is reached. Convergence for GBCD is assumed when gradient falls below 10^{-4}. Lowest test error among all methods shown in boldface.

	Training Time (second)				Variance Time/sample (second)				Root Mean Squared Error			
	SD	SR	MPGP	GBCD	SD	SR	MPGP	GBCD	SD	SR	MPGP	GBCD
Calhouse	3	48	514	221	0.02	0.02	0.02	19	0.514	0.477	0.475	**0.472**
Outaouais	5	61	627	5021	0.02	0.02	0.02	12	0.434	0.326	0.221	**0.122**
Kin40k	3	85	815	4233	0.02	0.02	0.02	48	0.386	0.164	0.113	**0.071**
Sarcos	4	142	1389	1879	0.02	0.02	0.02	31	0.169	0.128	0.121	**0.107**
Friedman1	3	231	2721	3301	0.02	0.02	0.02	75	0.031	0.015	0.012	**0.009**

Table 3: Average variance time of GBCD on each test sample and root mean square error of the predictive variance relative to that of Cholesky factorization. Variance time denotes the time of computing the predictive variance. Time is given in seconds. GBCD is stopped when the infinite norm of the gradient is smaller than 10^{-4}.

	Variance Time/sample	Relative RMSE
Calhouse	12	0.00004
Outaouais	6	0.001
Kin40k	20	0.02
Sarcos	6	0.002
Friedman1	4	0.001

quire fast testing speed. GBCD can be eventually used also in a sparse GPR setting with very large datasets–in this case even representative (sparse) subsets would be large.

3.3 Human Pose Estimation

This section presents our results on the HumanEva-1 dataset (Sigal & Black, 2006), a computer vision database that contains a number of human motion sequences of walking, jogging, throw-catch, gestures, and boxing (the backgrounds are known and fairly uniform, hence silhouettes can be computed). See (Bo et al., 2008; Bo & Sminchisescu, 2008; Sminchisescu et al., 2006) for alternative predictors, image representatons and results reported for this problem. The training set consists of pairs of human (image) silhouette-based descriptors and three-dimensional human poses (obtained using a motion capture system, that delivers synchronized images of people and corresponding information about their 3D pose) represented as 45d vectors of three-dimensional body joint positions (a 3d position per human body joint x 15 joints). All poses are preprocessed by subtracting the root joint location from the other joint centers for every frame. We use datasets corresponding to the same set of human motions, captured from three different cameras: C1, C2, C3. The human silhouettes are represented using histograms of semi-local shape contexts – a descriptor that encodes the spatial arrangement of edges around a given image location. To compute shape context descriptors (HistoSC), contours are extracted from the silhouette image and 400 points are sampled on its edges, both internal and external. The shape context descriptor at each point is computed based on 15 angular bins and 8 radial bins, surrounding it. The SC at each of the 400 sample points are accumulated over images subsampled from the training set (typically every 15) and used to generate a codebook using vector quantization. The codebook has 300 clusters obtained using k-means (hence the descriptor size is 300). The descriptor of a new human silhouette, e.g. for testing, is obtained by extracting shape contexts on the edges and vector-quantizing with respect to the existing codebook.

Table 5: Training time and mean joint error of MPGP and GBCD on HumanEva 1. Time is given in seconds and error given in mm, normalized per three-dimensional human body joint location.

	Training Time		Mean Joint Error	
	MPGP	GBCD	MPGP	GBCD
HistoSC	5237	3891	62.4	59.6

Training and test sets consist of 26572 and 8757 samples, respectively. The size of the subset of regressors is set to 2000 for MPGP. We report the mean joint error in table 5. As expected, our method is more accurate than sparse GPR models. The point is not that much to re-confirm an intuitive outcome, but to show that it is practical to obtain more accurate results on large datasets, otherwise only approachable using sparse methods.

4 Conclusions

We have presented an efficient variable decomposition algorithm that is capable of solving large scale Gaussian Process regression problems. The algorithm, referred to as GBCD converges to the global optimum of the objective function and can accurately handle large data sets of 100,000 training samples or more. We have shown that GBCD offers competitive solutions both in terms of accuracy and in terms of training/test time, and it compares favorably with dense and sparse GP methods on machine

learning and computer vision datasets. Although we have focused on Gaussian process regression, GBCD is potentially relevant for the solving kernel-based linear systems arising in other models or applications. The idea of breaking a large scale optimization into a series of smaller subproblems and then selecting an active set by objective-sensitive greedy algorithms is quite general. We hope that this strategy will be useful for other optimization problems.

Appendix

Proof of Theorem 1

From the running process of GBCD, we have

$$f\left(\boldsymbol{\alpha}^{k+1}\right) - f\left(\boldsymbol{\alpha}^{k}\right) = \frac{1}{2}\left(\Delta\boldsymbol{\alpha}_B^{opt}\right)^\top \bar{\mathbf{K}}_{BB}\Delta\boldsymbol{\alpha}_B^{opt} + \left(\mathbf{g}_B^k\right)^\top \Delta\boldsymbol{\alpha}_B^{opt} \tag{18}$$

Substituting $\bar{\mathbf{K}}_{BB}\Delta\boldsymbol{\alpha}_B^{opt}$ into (18), we have

$$f\left(\boldsymbol{\alpha}^{k+1}\right) - f\left(\boldsymbol{\alpha}^{k}\right) = -\frac{1}{2}\left(\Delta\boldsymbol{\alpha}_B^{opt}\right)^\top \bar{\mathbf{K}}_{BB}\Delta\boldsymbol{\alpha}_B^{opt} \tag{19}$$

Since $\bar{\mathbf{K}}_{BB}$ is a positive definite matrix, there exists one orthonormal matrix $\mathbf{U}$ such that $\bar{\mathbf{K}}_{BB} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$, where $\boldsymbol{\Lambda} = diag\left(\lambda_1(B), \cdots, \lambda_m(B)\right)$. Thus we have

$$\begin{aligned}\left(\Delta\boldsymbol{\alpha}_B^{opt}\right)^\top \bar{\mathbf{K}}_{BB}\Delta\boldsymbol{\alpha}_B^{opt} &= \left(\Delta\boldsymbol{\alpha}_B^{opt}\right)^\top \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top\Delta\boldsymbol{\alpha}_B^{opt} \\ &= \sum_{i=1}^{m}\lambda_i(B)\left(\mathbf{U}^\top\Delta\boldsymbol{\alpha}_B^{opt}\right)_i^2 \\ &\geq \lambda_{min}(B)\sum_{i=1}^{m}\left(\mathbf{U}^\top\Delta\boldsymbol{\alpha}_B^{opt}\right)_i^2\end{aligned} \tag{20}$$

where $\lambda_{min}(B) = \min_{1\leq i\leq m}\left(\lambda_i(B)\right)$. Since $\mathbf{U}$ is orthonormal, we have

$$\begin{aligned}\lambda_{min}(B)\sum_{i=1}^{m}\left(\mathbf{U}^\top\Delta\boldsymbol{\alpha}_B^{opt}\right)_i^2 &= \lambda_{min}(B)\left(\Delta\boldsymbol{\alpha}_B^{opt}\right)^\top \mathbf{U}\mathbf{U}^\top\Delta\boldsymbol{\alpha}_B^{opt} \\ &= \lambda_{min}(B)\left(\Delta\boldsymbol{\alpha}_B^{opt}\right)^\top \Delta\boldsymbol{\alpha}_B^{opt} \\ &\geq \lambda_{min}\left(\Delta\boldsymbol{\alpha}_B^{opt}\right)^\top \Delta\boldsymbol{\alpha}_B^{opt}\end{aligned} \tag{21}$$

Substituting (20) into (21), we get

$$\begin{aligned}f\left(\boldsymbol{\alpha}^{k+1}\right) - f\left(\boldsymbol{\alpha}^{k}\right) &\leq -\frac{1}{2}\lambda_{min}\left(\Delta\boldsymbol{\alpha}_B^{opt}\right)^\top \Delta\boldsymbol{\alpha}_B^{opt} \\ &= -\frac{1}{2}\lambda_{min}\left\|\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^{k}\right\|_2^2\end{aligned} \tag{22}$$

Proof of Theorem 2

The strictly positive definiteness of $\bar{\mathbf{K}}$ guarantees $\lambda_{min} > 0$. (22) implies that $\{f(\boldsymbol{\alpha}^k)\}$ is a decreasing sequence. Given that $f(\boldsymbol{\alpha}^k) \geq -\frac{1}{2}\mathbf{y}^\top\bar{\mathbf{K}}^{-1}\mathbf{y} > -\infty$, we have that $\lim_{k\to\infty}(f(\boldsymbol{\alpha}^k)) = -\frac{1}{2}\mathbf{y}^\top\bar{\mathbf{K}}^{-1}\mathbf{y}$. Applying (22) again, we obtain that $\{\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k\}$ converges to 0.

Since $f(\boldsymbol{\alpha})$ is a positive definite quadratic form, the set $\{\boldsymbol{\alpha} | f(\boldsymbol{\alpha}) \geq f(\boldsymbol{\alpha}^0)\}$ is compact. $\{\boldsymbol{\alpha}^k\}$ lies in this set, so it is a bounded sequence. Let $\bar{\boldsymbol{\alpha}}$ be the limit of any convergent subsequence $\{\boldsymbol{\alpha}^k\}$, $k \in \Gamma$. Since there are only a finite number of variables, there exists at least one active set B^* which occurs infinitely in this subsequence. Let $\Gamma^* \subseteq \Gamma$ be the set of superscripts corresponding to B^* and s_1 the index in B^* first selected by GBCD. The s_1-th component of the gradient vector at $\bar{\boldsymbol{\alpha}}$ is

$$\begin{aligned}g_{s_1}(\bar{\boldsymbol{\alpha}}) &= \lim_{k\to\infty, k\in\Gamma^*} g_{s_1}\left(\boldsymbol{\alpha}^k\right) \\ &= \lim_{k\to\infty, k\in\Gamma^*} g_{s_1}\left(\boldsymbol{\alpha}^k\right) - g_{s_1}\left(\boldsymbol{\alpha}^{k+1}\right) + \lim_{k\to\infty, k\in\Gamma^*} g_{s_1}\left(\boldsymbol{\alpha}^{k+1}\right)\end{aligned} \tag{23}$$

Given $B(k)$, the active set at the k-th iteration, we have

$$\mathbf{g}_{B(k)}\left(\boldsymbol{\alpha}^{k+1}\right) = \mathbf{g}_{B(k)}\left(\boldsymbol{\alpha}^{k}\right) + \bar{\mathbf{K}}_{B(k)B(k)}\Delta\boldsymbol{\alpha}_{B(k)}^{opt} \tag{24}$$

Substituting $\Delta\boldsymbol{\alpha}_{B(k)}^{opt} = -\bar{\mathbf{K}}_{B(k)B(k)}^{-1}\mathbf{g}_{B(k)}^k$ into (24), we get $\mathbf{g}_{B(k)}(\boldsymbol{\alpha}^{k+1}) = \mathbf{0}$. This indicates that $\lim_{k\to\infty, k\in\Gamma^*}(g_{s_1}(\boldsymbol{\alpha}^{k+1})) = 0$. Since $\{\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k\}$ converges to 0, $\lim_{k\to\infty, k\in\Gamma^*}(g_{s_1}(\boldsymbol{\alpha}^k) - g_{s_1}(\boldsymbol{\alpha}^{k+1})) = 0$. Thus, we obtain

$$g_{s_1}(\bar{\boldsymbol{\alpha}}) = 0 \tag{25}$$

From step 3 of the greedy approximation (we search exhaustively for the first active variable), we have

$$\frac{-g_{s_1}\left(\boldsymbol{\alpha}^k\right)^2}{2\left(k(\mathbf{x}_{s_1}, \mathbf{x}_{s_1}) + \sigma^2\right)} \leq \frac{-g_i\left(\boldsymbol{\alpha}^k\right)^2}{2\left(k(\mathbf{x}_i, \mathbf{x}_i) + \sigma^2\right)} \quad \forall i \in \{1, \cdots, n\} \tag{26}$$

Because $\bar{\mathbf{K}}$ is a positive definite matrix, $k(\mathbf{x}_i, \mathbf{x}_i) > 0$, $\forall i \in \{1, \cdots, n\}$. Taking the limit of (26), we get

$$\begin{aligned}g_i(\bar{\boldsymbol{\alpha}})^2 &= \left[\lim_{k\to\infty, k\in\Gamma^*} g_i\left(\boldsymbol{\alpha}^k\right)\right]^2 \\ &\leq \frac{k(\mathbf{x}_i, \mathbf{x}_i) + \sigma^2}{k(\mathbf{x}_{s_1}, \mathbf{x}_{s_1}) + \sigma^2}\left[\lim_{k\to\infty, k\in\Gamma^*} g_{s_1}\left(\boldsymbol{\alpha}^k\right)\right]^2 \\ &= \frac{k(\mathbf{x}_i, \mathbf{x}_i) + \sigma^2}{k(\mathbf{x}_{s_1}, \mathbf{x}_{s_1}) + \sigma^2}\left[g_{s_1}(\bar{\boldsymbol{\alpha}})\right]^2 \\ &= 0 \quad \forall i \in \{1, \cdots, n\}\end{aligned} \tag{27}$$

Thus, $\bar{\boldsymbol{\alpha}}$ is the optimal solution of (3). Strict convexity of $f(\boldsymbol{\alpha})$ further implies that the sequence $\{\boldsymbol{\alpha}^k\}$ itself converges to the global optimum of (3).

Acknowledgements: This work was supported, in part, by the EC and the NSF, under awards MCEXT-025481 and IIS-0535140.

References

Beatson, R. K., Light, W. A., & Billings, S. (2000). Fast solution of the radial basis function interpolation equations: Domain decomposition methods. *SIAM J. Sci. Comput.*, *22*, 1717–1740.

Bertsekas, D. (1999). *Nonlinear programming, 2nd ed.* Belmont, MA: Athena Scientific.

Bo, L., & Sminchisescu, C. (2008). Twin Gaussian Processes for Structured Prediction. *Snowbird Learning*.

Bo, L., Sminchisescu, C., Kanaujia, A., & Metaxas, D. (2008). Fast Algorithms for Large Scale Conditional 3D Prediction. *IEEE Conference on Computer Vision and Pattern Recognition*.

Chang, C.-C., & Lin, C.-J. (2001). *Libsvm: a library for support vector machines* (Technical Report). Sofware available at http://www.csie.ntu.edu.tw/cjlin/libsvm.

Csató, L., & Opper, M. (2002). Sparse on-line gaussian processes. *Neural Computation*, *14*, 641–668.

Friedman, J. (1991). Multivariate adaptive regression splines. *Annals of Statistics*, *19*, 1–141.

Joachims, T. (1999). Making large-scale svm learning practical. *Advances in Kernel Methods - Support Vector Learning*.

Keerthi, S., & Chu, W. (2006). A matching pursuit approach to sparse gaussian process regression. In Y. Weiss, B. Schölkopf and J. Platt (Eds.), *Advances in neural information processing systems 18*, 643–650. Cambridge, MA: MIT Press.

Keerthi, S. S., & Shevade, S. K. (2003). Smo for least squares svm formulations. *Neural Computation*, *15*, 487–507.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., & Murthy, K. R. K. (2001). Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, *13*, 637–649.

Keerthi, S. S., Sindhwani, V., & Chapelle, O. (2006). An efficient method for gradient-based adaptation of hyperparameters in svm models. *Advances in Neural Information Processing System 19*.

Lawrence, N., Seeger, M., & Herbrich, R. (2003). Fast sparse Gaussian process methods: the informative vector machine. *Advances in Neural Information Processing System 15*.

Li, W., Lee, K.-H., & Leung, K.-S. (2007). Large-scale rlsc learning without agony. *ICML '07: Proceedings of the 24th international conference on Machine learning* (pp. 529–536). New York, NY, USA: ACM.

Ortega, J., & Rheinboldt, W. (1970). *Iterative solution of nonlinear equations in several variables*. New York: Academic Press.

Platt, J. (1999). Sequential minimal optimization: a fast algorithm for training support vector machines. *Advance in Kernel Methods - Support Vector Learning*.

Quiñonero-Candela, J., & Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, *6*, 1935–1959.

Rasmussen, C. E., & Candela, J. Q. (2005). Healing the relevance vector machine through augmentation. *Proceedings of the 22nd International Conference on Machine Learning*.

Rasmussen, C. E., & Williams, C. K. (2006). *Gaussian processes for machine learning*. Adaptive Computation and Machine Learning. The MIT Press.

Seeger, M., Williams, C. K. I., & Lawrence, N. (2003). Fast forward selection to speed up sparse gaussian process regression. *Ninth International Workshop on Artificial Intelligence and Statistics*.

Shen, Y., Ng, A., & Seeger, M. (2006). Fast gaussian process regression using kd-trees. In Y. Weiss, B. Schölkopf and J. Platt (Eds.), *Advances in neural information processing systems 18*, 1225–1232. Cambridge, MA: MIT Press.

Sigal, L., & Black, M. (2006). *HumanEva: Synchronized Video and Motion Capture Dataset for Evaluation of Articulated Human Motion* (Technical Report CS-06-08). Brown University.

Sminchisescu, C., Kanaujia, A., & Metaxas, D. (2006). Learning Joint Top-down and Bottom-up Processes for 3D Visual Inference. *IEEE Conference on Computer Vision and Pattern Recognition*.

Smola, A. J., & Bartlett, P. L. (2001). Sparse greedy gaussian process regression. *Advances in Neural Information Processing System 13*.

Smola, A. J., & Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. *Proceedings of the 17th International Conference on Machine Learning*.

Stoer, J., & Bulirsch, R. (1993). *Introduction to numerical analysis*. New York: Springer-Verlag.

Vincent, P., & Bengio, Y. (2002). Kernel matching pursuit. *Machine Learning*, *48*, 165–187.

Williams, C. K. I., & Rasmussen, C. E. (1996). Gaussian processes for regression. *Advances in Neural Information Processing System 8*.

Yang, C., Duraiswami, R., & Davis, L. (2005). Efficient kernel machines using the improved fast gauss transform. In L. K. Saul, Y. Weiss and L. Bottou (Eds.), *Advances in neural information processing systems 17*, 1561–1568. Cambridge, MA: MIT Press.

Zoutendijk, G. (1970). *Methods of feasible directions: a study in linear and non-linear programming*. Elsevier.

CORL: A Continuous-state Offset-dynamics Reinforcement Learner

Emma Brunskill* **Bethany R. Leffler†** **Lihong Li†** **Michael L. Littman†** **Nicholas Roy***
* Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02143
† Department of Computer Science
Rutgers University
Piscataway, NJ 08854

Abstract

Continuous state spaces and stochastic, switching dynamics characterize a number of rich, real-world domains, such as robot navigation across varying terrain. We describe a reinforcement-learning algorithm for learning in these domains and prove for certain environments the algorithm is probably approximately correct with a sample complexity that scales polynomially with the state-space dimension. Unfortunately, no optimal planning techniques exist in general for such problems; instead we use fitted value iteration to solve the learned MDP, and include the error due to approximate planning in our bounds. Finally, we report an experiment using a robotic car driving over varying terrain to demonstrate that these dynamics representations adequately capture real-world dynamics and that our algorithm can be used to efficiently solve such problems.

1 INTRODUCTION

Reinforcement learning (RL) has had some impressive successes, such as model helicopter flying (Ng et al., 2004) and expert software backgammon players (Tesauro, 1994). Two key challenges in reinforcement learning are scaling to large worlds, which often involves a form of generalization, and efficiently handling the exploration/exploitation tradeoff. Many real-life problems involve real-valued state variables: discretizing such environments causes an exponential growth in the number of states as the state dimensionality increases, and so solutions that directly reason with continuous-states are of important consideration.

In this paper, we build on recent work on probably efficient reinforcement learning (Kearns & Singh, 2002; Brafman & Tennenholtz, 2002; Strehl et al., 2006) and focus on continuous-state, discrete-action environments. We consider the case when the dynamics can be described as switching noisy offsets where the parameters of the dynamics depend on the state's "type" t and the action taken a. More formally,

$$s' = s + \beta_{at} + \varepsilon_{at}, \quad (1)$$

where s is the current state, s' is the next state, $\varepsilon_{at} \sim \mathcal{N}(0, \Sigma_{at})$ is drawn from a zero-mean Gaussian with covariance Σ_{at} and β_{at} is the offset.

An example where we expect such dynamics to arise is during autonomous traversal of varying terrain. Here, types represent the ground surface, such as dirt or rocks. The dynamics of the car may be approximated by an offset from the prior state plus some noise, where the offset and noise depend on the surface underneath the car. These models could be useful approximations in a number of other problems, including transportation planning (learning the mean speed and variance of interstate highways and local streets for path planning to a goal location), and packet routing (learning that wireless and ethernet have different bandwidth/usage patterns and routing accordingly).

We present a new RL algorithm for learning in continuous-state, discrete-action Markov decision processes (MDPs) with switching noisy offset dynamics and show that this algorithm is probably approximately correct (PAC) in certain environments with a sample complexity that scales polynomially with the state space dimensionality. We perform planning using fitted value iteration (FVI) and incorporate the error due to approximate planning into our bounds.

Finally, we present experiments on a small robot task that involves navigation over varying terrains. These experiments demonstrate that our dynamics models can adequately capture real-life dynamics and our algorithm can quickly learn good policies in such environments.

2 A CONTINUOUS-STATE OFFSET-DYNAMICS REINFORCEMENT LEARNER

This section introduces terminology and then presents our algorithm.

2.1 BACKGROUND

The world is characterized by a continuous-state discounted MDP $M = \langle S, A, p(s'|s,a), R, \gamma \rangle$ where $S \subseteq \mathbb{R}^{N_{dim}}$ is the N_{dim}-dimensional state space, A is a set of discrete actions, $p(s'|s,a)$ is the unknown transition dynamics that satisfy the parametric form of Equation 1, $\gamma \in [0,1)$ is the discount factor and $R : S \times A \rightarrow [0,1]$ is the known reward model. In addition to the standard MDP formulation, each state s is associated with a single observable type $t \in T$. The total number of types is N_T. The dynamics of the environment are determined by the current state type t and action a taken:

$$p(s'|s,a) = \mathcal{N}(s'|s + \beta_{at}, \Sigma_{at}). \quad (2)$$

In other words, types partition the state space into regions, and each region is associated with particular pair of dynamics parameters.

In this work, we focus on the known reward model, unknown dynamics model situation. The parameters of the dynamics model, β_{at} and Σ_{at}, are assumed to be unknown for all types t and actions a at the start of learning. This model is a departure from prior related work (Abbeel & Ng, 2005; Strehl & Littman, 2008), which focuses on a more general linear dynamics model but assumes a single type and that the variance of the noise Σ_{at} is known. We argue there exist interesting problems where the variance of the noise is unknown and estimating this noise may provide the key distinction between the dynamics models of different types.

In reinforcement learning, the agent must learn to select an action a based on its current state s. At each time step, it receives an immediate reward r also based on its current state[1]. The agent then moves to a next state s' according to the dynamics model. The goal is to learn a policy $\pi : S \rightarrow A$ that allows the agent to choose actions to maximize the total rewards it receives. The value of a particular policy is the expected discounted sum of future rewards that will be received from following this policy, and is denoted $V^{\pi}(s) = E_{\pi}[\sum_{j=0}^{\infty} \gamma^j r_j | s_0 = s]$, where r_j is the reward received on the j-th time step and s_0 is the initial state of the agent. Let π^* be the optimal policy, and its associated value function be $V^*(s)$.

2.2 ALGORITHM

Our algorithm (*c.f.*, Algorithm 1) is derived from the R-max algorithm of Brafman and Tennenholtz (2002). We first form a set of $\langle t, a \rangle$ tuples, one for each type–action pair. Note that each tuple corresponds to a particular pair of dynamics model parameters, $\langle \beta_{at}, \Sigma_{at} \rangle$. A tuple is considered to be "known" if the agent has been in type t and taken action a a number N_{at} times. At each timestep, we construct a new MDP $\hat{M}_K$ as follows. If the number of times a tuple has been experienced, n_{at}, is greater than or equal to N_{at}, then we estimate the parameters for this dynamics model using maximum-likelihood estimation:

$$\tilde{\beta}_{at} = \frac{\sum_{i=1}^{n_{at}} (s'_i - s_i)}{n_{at}} \quad (3)$$

$$\tilde{\Sigma}_{at} = \frac{\sum_{i=1}^{n_{at}} (s'_i - s_i - \tilde{\beta}_{at})(s'_i - s_i - \tilde{\beta}_{at})^T}{n_{at}} \quad (4)$$

where the sum ranges over all state action pairs experienced for which the type of s_i was t and the action taken was a.

Otherwise, we set the dynamics model for all states and action associated with this type–action tuple to be a transition with probability 1 back to the same state. We also modify the reward function for all states associated with an unknown type–action tuple $\langle t_u, a_u \rangle$ so that all state–action values $Q(s_{t_u}, a_u)$ have a reward of $V_{\max}$ (the maximum value possible, $1/(1-\gamma)$). We then seek to solve $\hat{M}_K$. This MDP includes switching dynamics with continuous states, and we are aware of no exact optimal planners for such MDPs[2]. Instead, we will use fitted value iteration to approximately solve the MDP.

In FVI, the value function is represented explicitly at only a fixed set of states that are (for example) uniformly spaced in a grid over the state space. Planning requires performing Bellman backups for each grid point μ_f. Since we are *only* performing backups of the value function at a set of grid points μ_f, we need a function approximator to estimate the

Algorithm 1 CORL

1: **Input:** N_A, N_{dim}, N_T, R, $\Sigma_{\max}$, $\Sigma_{\min}$, γ, ϵ, and δ.
2: Set all type–action tuples $\langle t, a \rangle$ to be unknown and initialize the dynamics models (see text) to create an empirical known-state MDP model $\hat{M}_K$.
3: Select a fixed set of evenly spaced points for fitted value iteration.
4: Start in a state s_0.
5: **loop**
6: Solve MDP $\hat{M}_K$ using fitted value iteration and denote its optimal value function by Q_t.
7: Select action $a = \text{argmax}_a \, Q_t(s,a)$.
8: Transition to the next state s'.
9: Increment the appropriate n_{at} count (where t is the type of state s) given the observed transition tuple $\langle s, a, s' \rangle$.
10: If n_{at} exceeds N_{at} where N_{at} is specified according to the analysis, then mark $\langle a, t \rangle$ as "known" and estimate the dynamics model parameters for this tuple.
11: **end loop**

[1]For simplicity, the reward is assumed to be only a function of state in this paper, but the arguments can be easily extended to where the reward model is also a function of the action chosen.

[2]In contrast, optimal control is possible for linear Gaussian (non-switching) dynamics continuous-state systems with linear quadratic reward functions.

value of other points that are not in this fixed set. We can use Gaussian kernel functions to interpolate the value at the grid points to other points. The value of a state s is

$$V(s) = \max_a \sum_{f=1}^{F} w_f \mathcal{N}(s; \mu_f, \Sigma_f) Q(\mu_f, a), \quad (5)$$

where w_f is a scalar and $\mathcal{N}(s; \mu_f, \Sigma_f)$ represents a Gaussian with mean at grid point μ_f and variance Σ_f evaluated at state s. The grid-point locations, variances and weights (μ_f, Σ_f, w_f) are defined so

$$\sum_{f=1}^{F} w_f \mathcal{N}(s; \mu_f, \Sigma_f) \approx 1 \quad (6)$$

for all states s of interest. We would like this expression to exactly equal 1 for all states of interest as that guarantees the function approximator is an averager and therefore discounted infinite horizon fitted value iteration is guaranteed to converge (Gordon, 1995). In practice, if Gaussians are placed at uniform intervals over the state space of interest, then this expression can be extremely close to 1. Indeed, as long as the sum in Equation 6 sums to less than or equal to 1 for all states, then the approximator operator is guaranteed to be a non-expansion in the max norm and therefore discounted infinite horizon fitted value iteration is still guaranteed to converge.

Substituting this representation of the value function in place of $V(s')$ and using the dynamics model in the Bellman backup equation, we can perform the integration over future reward in closed form to get

$$\begin{aligned} V(\mu_\cdot) \quad = \quad & R(\mu_\cdot) + \gamma \max_a \sum_f w_f \cdot \\ & \mathcal{N}(\mu_f; \mu_\cdot + \beta_{at_f}, \Sigma_{at_f} + \Sigma_f) V(\mu_f). \end{aligned}$$

For a given basis set of fixed states μ_f, the majority of the right side can be computed once and used repeatedly during value iteration; essentially, the continuous-state MDP is converted to a new discrete-state MDP where the states are the fixed points.

At each timestep, the agent chooses the action that maximizes the estimate of its current value according to Q_t: $a = \text{argmax}_a Q_t(s, a)$. The complete algorithm is shown in Algorithm 1.

3 LEARNING COMPLEXITY

In Section 4, we will analyze our algorithm in a family of MDPs with switching noisy offsets, and examine how many samples N_{at} are necessary in order to produce a good policy. In particular, we prove it is probably approximately correct with a sample complexity (N_{at}) that scales polynomially with the number of dimensions in the state space.

When analyzing the performance of an RL algorithm $\mathcal{A}$, there are many potential criteria to use. In our work, we will focus predominantly on sample complexity with a brief mention of computational complexity. Computational complexity refers to the number of operations executed by the algorithm for each step taken by the agent in the environment. We will follow Kakade (2003) and use *sample complexity* as shorthand for the *sample complexity of learning*. It is the number of timesteps at which the algorithm, when viewed as a non-stationary policy π, is not ϵ-optimal at the current state; that is, $Q^*(s, a) - Q^\pi(s, a) > \epsilon$ where Q^* is the optimal state-action value function and Q^π is the state–action value function of the non-stationary policy π. Following Strehl et al. (2006), we are interested in showing, for a given ϵ and δ, that with probability at least $1 - \delta$ the sample complexity of the algorithm is less than or equal to a polynomial function of MDP parameters. Note that we only consider the number of samples to ensure the algorithm will learn and execute a near-optimal policy with high probability. As the agent acts in the world, it may be unlucky and experience a series of state transitions that poorly reflect the true dynamics, due to noise.

We will follow the lead of a recent and related continuous-state reinforcement-learning algorithm by Strehl and Littman (2008), and use the framework of Strehl et al. (2006). Strehl et al. (2006) defined an algorithm to be greedy if it chooses its action to be the one that maximizes the value of the current state s ($a = \text{argmax}_{a \in A} Q(s, a)$). Their paper's main result goes as follows: let $\mathcal{A}(\epsilon, \delta)$ denote a greedy learning algorithm. Maintain a list K of "known" state–action pairs. At each new timestep, this list stays the same unless during that timestep a new state–action pair becomes known. MDP M_K is the known state–action MDP (where the construction is essentially the same as described earlier, except that the reward and transition functions are the same as the original MDP for known state–action pairs) and π is the greedy policy with respect to the current value function, Q_t. Assume that ϵ and δ are given and the following 3 conditions hold for all states, actions and timesteps:

1. $Q^*(s, a) - Q_t(s, a) \leq \epsilon$.
2. $V_t(s) - V^{\pi_t}_{M_K}(s) \leq \epsilon$.
3. The total number of times the agent visits a state–action tuple that is not in K is bounded by $\zeta(\epsilon, \delta)$ (the *learning complexity*).

Then, Strehl et al. (2006) show on any MDP M, $\mathcal{A}(\epsilon, \delta)$ will follow a 4ϵ-optimal policy from its initial state on all but N_{total} timesteps with probability at least $1 - 2\delta$, where N_{total} is a polynomial in the problem's parameters $(\zeta(\epsilon, \delta), \frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{1-\gamma})$.

The majority of our analysis will focus on showing that our algorithm fulfills these three criteria. In our approach,

we will define the known state–action pairs to be all those state–actions for which the type–action pair $\langle t(s), a\rangle$ is known.

Before we commence, we first briefly give some intuition for the above three criteria and describe how we will proceed in proving our algorithm satisfies them. Together, the first and second criteria can be interpreted as saying that the algorithm should produce accurate value estimates of the all state-action pairs in the known MDP, and that it should be optimistic about the values of all state–action pairs. The first criterion is more challenging to demonstrate. To show our estimates of known state–action pairs are close to their real values, we must consider two potential sources of error that could prevent it. The first is that the model dynamics are only estimated from the samples experienced, and so the dynamics model estimates may deviate from the true dynamics. In Proposition 4.1 and Lemmas 4.2, 4.3, and 4.4, we bound the number of samples necessary to ensure the dynamics model parameter estimates are close to the true dynamics. The second source of error comes from solving the MDP. We cannot currently perform exact optimal planning for these continuous-state noisy offset MDPs, and therefore we use approximate planning. In Section 4.2, we bound the error it introduces. We then combine these results in Lemma 4.5 to bound the error between our estimate of the value of the known-state MDP and the true optimal values. Theorem 4.6 uses this result to prove the algorithm is probably approximately correct with a sample complexity that scales polynomially in the problem parameters, including the state-space dimension.

Note that our use of an approximate planner is a departure from most related work on PAC RL. Existing work typically assumes the existence of a planning oracle for choosing actions given the estimated model.

To ensure fitted value iteration produces highly accurate results, our algorithm's worst-case computational complexity is exponential in the number of state dimensions. While this fact prevents it from being theoretically computationally efficient, our experimental results demonstrate our algorithm performs well compared to related approaches in a real-life robot problem.

4 ANALYSIS

This section provides a formal analysis of Algorithm 1. For simplicity, it assumes a diagonal covariance matrix for the noise model: $\Sigma = \text{diag}(\sigma_1^2, \sigma_2^2, \cdots, \sigma_{N_{dim}}^2)$. We believe it is possible to extend the analysis to the general covariance matrices, and leave it for future work. We also assume that the absolute values of the components in β_{at} and Σ_{at} are upper bounded by some known constants, B_β and B_σ, respectively. This assumption is often true in practice. We denote by $|D|$ the determinant of matrix D. Due to space limitations some details will be omited in our analysis: please see Brunskill et al. (2008) for full proofs.

4.1 MODEL ACCURACY

We first establish the distance between two dynamics models with different parameters. It will be important for analyzing the potential difference in expected received reward between a MDP with the true dynamics model and an estimated (from the data) dynamics model. Following Abbeel and Ng (2005), we use the variational distance

$$d_{var}(P(x), Q(x)) = \frac{1}{2}\int_{\mathcal{X}} |P(x) - Q(x)| dx. \quad (7)$$

Proposition 4.1 *Assume that both Σ_1 and Σ_2 are diagonal matrices and let $\sigma_{\min}$ be the minimum standard deviation along any of the dimensions. Also, assume without loss of generality that $|\Sigma_1| \leq |\Sigma_2|$. Then,*

$$\begin{aligned} & d_{var}(\mathcal{N}(s'|\beta_1 + s, \Sigma_1), \mathcal{N}(s'|\beta_2 + s, \Sigma_2)) \\ & \leq 1 - \left(\prod_{i=1}^{N_{dim}} \frac{\min[\sigma_{1i}^2, \sigma_{2i}^2]}{\sigma_{2i}^2} \right)^{0.5} + \frac{||\beta_2 - \beta_1||_2}{\sqrt{(2\pi)}\sigma_{\min}}, \end{aligned}$$

where σ_{ki}^2 is the i-th diagonal component of Σ_k.

Proof

$$\begin{aligned} & d_{var}(\mathcal{N}(s'|\beta_1 + s, \Sigma_1), \mathcal{N}(s'|\beta_2 + s, \Sigma_2)) \\ & = \frac{1}{2}\int_{s'} |\mathcal{N}(s'|\beta_1 + s, \Sigma_1) - \mathcal{N}(s'|\beta_2 + s, \Sigma_2)| ds' \\ & = \frac{1}{2}\int_{s'} |\mathcal{N}(s'|\beta_1 + s, \Sigma_1) - \mathcal{N}(s'|\beta_2 + s, \Sigma_1) + \\ & \quad \mathcal{N}(s'|\beta_2 + s, \Sigma_1) - \mathcal{N}(s'|\beta_2 + s, \Sigma_2)| ds', \end{aligned}$$

where we have simply added and subtracted the same term. Using the triangle inequality, we can split the expression into two terms:

$$\begin{aligned} & d_{var}(\mathcal{N}(s'|\beta_1 + s, \Sigma_1), \mathcal{N}(s'|\beta_2 + s, \Sigma_2)) \\ & \leq \frac{1}{2}\int_{s'} |\mathcal{N}(s'|\beta_1 + s, \Sigma_1) - \mathcal{N}(s'|\beta_2 + s, \Sigma_1)| \, ds' \\ & + \frac{1}{2}\int_{s'} |\mathcal{N}(s'|\beta_2 + s, \Sigma_1) - \mathcal{N}(s'|\beta_2 + s, \Sigma_2)| \, ds', \end{aligned}$$

one where the means are the same and the variances are different, and one where the variances are the same and the means are different. The second term is summing all the area between the lines defining the two Gaussians (which are centered at the same mean). An alternate way to think about computing this area is to take the sum of the area under the two Gaussians (which is simply 2) and subtract off two times the area of the intersection, D, between them:

$$\frac{1}{2}(2 - 2D) = 1 - D. \quad (8)$$

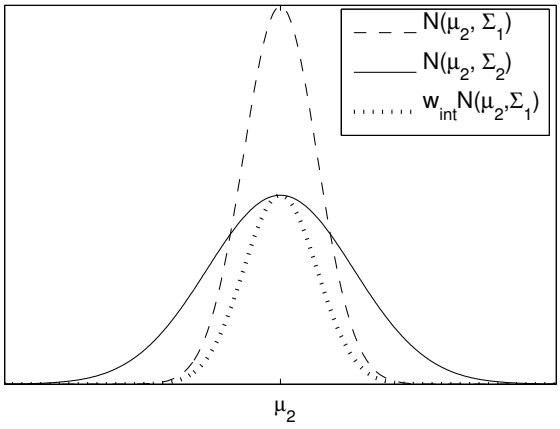


Figure 1: Two Gaussians with identical means and different variances, and a new weighted Gaussian that lies entirely inside their intersection.

To upperbound this term, we would like to find a lower bound on the area of intersection between these two Gaussians, D. We can construct a new weighted Gaussian that lies entirely within the intersection area and has the same mean as the two Gaussians ($\beta_2 + s$) (see Figure 1 for a one-dimensional example). We can set the covariance of this new Gaussian by setting its variance along each dimension i to be the smaller of the two Gaussians' variances: $\sigma^2_{int,i} = \min[\sigma^2_{i1}, \sigma^2_{i2}]$. We then determine the weight on the Gaussian w_{int} by requiring that its height at the mean be no more than the smaller of the two Gaussians. Since we have assumed that $|\Sigma_1| \leq |\Sigma_2|$, then the height at the mean of the smaller Gaussian is simply $1/((2\pi)^{N_{dim}/2}|\Sigma_2|^{0.5})$. Therefore, we can set w_{int} as

$$w_{int}\mathcal{N}_{int}(0, \Sigma_{int}) = \frac{1}{(2\pi)^{N_{dim}/2}|\Sigma_2|^{0.5}}.$$

Solving for w_{int}, we get

$$w_{int} = \left(\prod_{i=1}^{N_{dim}} \frac{\min[\sigma^2_{1i}, \sigma^2_{2i}]}{\sigma^2_{2i}} \right)^{0.5}.$$

This weighted Gaussian always lies within the intersection region and therefore the D is at least

$$\int w_{int}\mathcal{N}_{int}(s'|\beta_2 + s, \Sigma_{int})ds' = w_{int}.$$

Substituting this expression back into Equation 8,

$$\frac{1}{2}\int_{s'} |\mathcal{N}(s'|\beta_2 + s, \Sigma_1) - \mathcal{N}(s'|\beta_2 + s, \Sigma_2)|\, ds'$$
$$\leq \quad 1 - w_{int} \;=\; 1 - \left(\prod_{i=1}^{N_{dim}} \frac{\min[\sigma^2_{1i}, \sigma^2_{2i}]}{\sigma^2_{2i}} \right)^{0.5}. \quad (9)$$

Next, consider the first term in Equation 8, which looks at the difference between two Gaussians with different means and identical variances. From Abbeel and Ng (2005) (Proposition 7), this expression is upperbounded by

$$\frac{||\beta_2 + s - (\beta_1 + s)||_2}{\sqrt{2\pi}\sigma_{\min}} = \frac{||\beta_2 + \beta_1||_2}{\sqrt{2\pi}\sigma_{\min}}. \quad (10)$$

Combining Equations 10 and 9 gives the desired result. □

Note this function is 0 when the means and the variances are the same, as one would hope.[3]

We next seek to determine the number of samples necessary to ensure that d_{var} is tightly bounded when evaluated at the estimated model parameters and the true model parameters. Let us first define "good" samples as those for which $\|s' - s\|_\infty < B$ for some given $B > 0$. The value of B will be specified later.

Lemma 4.2 *Given any $\epsilon, \delta > 0$, define $T_\beta = \frac{2N_{dim}B^2}{\epsilon^2} \ln \frac{6N_{dim}}{\delta}$. If there are T_β good transition samples (s, a, s'), then with probability at least $1 - \frac{\delta}{3}$, the estimated offset parameter $\tilde{\beta}$, computed by Equation 3, deviates from the true offset parameter β^* by at most ϵ; formally, $\Pr(\|\tilde{\beta} - \beta^*\|_2 \leq \epsilon) \geq 1 - \frac{\delta}{3}$.*

Proof : (Sketch) The proof rests on an application of Hoeffding's inequality (Hoeffding, 1963). □

We next analyze the number of samples needed to estimate the variance accurately.

Lemma 4.3 *Assume $\|\tilde{\beta} - \beta\|_2 \leq \epsilon$. Given any $\epsilon, \delta > 0$, define $T_\sigma = \frac{8B^4}{\epsilon - \epsilon^2} \ln \frac{6N_{dim}}{\delta}$. If there are T_σ good transition samples (s, a, s'), then with probability at least $1 - \frac{\delta}{3}$, the estimated variance parameter $\tilde{\sigma}^2_i$, computed by Equation 4, deviates from the true variance parameter σ^2_i by at most ϵ for every dimension i; formally, $\Pr(\max_i |\tilde{\sigma}^2_i - \sigma^2_i| \leq \epsilon) \geq 1 - \frac{\delta}{3}$.*

Proof : (Sketch) The proof first relates the estimate of the variance using the current estimate of the offset parameter β to the variance around the true offset parameter, and then bounds this error using Hoeffding's inequality and a union bound. □

These two lemmas provide us with an estimate of how many good samples are necessary to achieve, with high probability, accurate estimates of the dynamics model parameters for every type–action pair. One additional lemma is needed to bound how many samples must be collected until enough such good samples are obtained.

[3]The true d_{var} is upper bounded by 1, whereas this expression can go higher, so it is overly pessimistic when the difference between the two Gaussians' parameters is large, but increasingly accurate as their difference goes to 0. Since we need to estimate the parameters fairly precisely, we are more concerned with this second case.

Lemma 4.4 *Let T be the number of observed samples before $T_0 = \max\{T_\beta, T_\sigma\}$ good samples are collected. Then,* $\Pr(T > \frac{\delta T_0}{\delta - 3N_{dim}p_0}) < \frac{\delta}{3}$, *where* $p_0 = \sqrt{\frac{8}{\pi}}\frac{B_\sigma^3}{(B-B_\beta)^3}$. *Here, setting* $B > B_\beta + \sqrt[6]{\frac{72N_{dim}^2}{\pi\delta^2}}B_\sigma$ *ensures* $\delta > 3N_{dim}p_0$.

Proof : It follows from a union bound that $\Pr(\|s' - s\|_\infty > B) \leq N_{dim}\Pr(|s_i' - s_i| > B)$ for all i. We will show that $\Pr(|s_i' - s_i| > B)$ is small. Let $\varphi(x)$ and $\Phi(x)$ be the probability density function and cumulative distribution function of the standard Gaussian distribution, respectively. Then,

$$\begin{aligned}\Pr(s_i' - s_i > B) &= 1 - \Phi\left(\frac{B-\beta_i^*}{\sigma_i}\right) \\ &\leq \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(B-\beta_i^*)^2}{2\sigma_i^2}\right)\frac{1}{\frac{B-\beta_i^*}{\sigma_i}} \\ &= \frac{\sigma_i}{\sqrt{2\pi}(B-\beta_i^*)}\exp\left(-\frac{(B-\beta_i^*)^2}{2\sigma_i^2}\right),\end{aligned}$$

where the first equality follows from the definition, and the inequality follows from the fact that $1-\Phi(y) < \frac{\varphi(y)}{y}$ when $y > 0$. Now, we can apply the inequality $e^{-x} < \frac{1}{1+x}$ to obtain

$$\begin{aligned}\Pr(s_i' - s_i > B) &\leq \frac{\sigma_i}{\sqrt{2\pi}(B-\beta_i^*)}\cdot\frac{1}{1+\frac{(B-\beta_i^*)^2}{2\sigma_i^2}} \\ &< \sqrt{\frac{2}{\pi}}\frac{\sigma_i^3}{(B-\beta_i^*)^3} \leq \sqrt{\frac{2}{\pi}}\frac{B_\sigma^3}{(B-B_\beta)^3}.\end{aligned}$$

Similarly, we may upperbound $\Pr(s_i' - s_i < -B)$ and thus, $\Pr(|s_i' - s_i| > B) < p_0$, where p_0 is given in the lemma statement.

Now, return to the full multivariate case:

$$\Pr(\|s' - s\|_\infty > B) \leq N_{dim}p_0 = \sqrt{\frac{8}{\pi}}\frac{B_\sigma^3 N_{dim}}{(B-B_\beta)^3}.$$

This inequality indicates that every sample is a "bad" sample with probability at most $N_{dim}p_0$. Given T i.i.d. samples, let $N(T)$ be the number of bad samples. Our estimation algorithm fails to have T_0 good samples if and only if $N(T) > T - T_0$. By Markov's inequality,

$$\Pr(N(T) > T - T_0) \leq \frac{\mathbf{E}[N(T)]}{T-T_0} < \frac{TN_{dim}p_0}{T-T_0}.$$

Solving for T by letting the last expression equal $\frac{\delta}{3}$ gives $T = \frac{\delta T_0}{\delta - 3N_{dim}p_0}$. We can obtain the minimum value of B by solving $3N_{dim}p_0 = \delta$ for B. □

Combining these results with Lemmas 4.2 and 4.3 gives a condition on the minimum number of samples necessary to ensure, with high probability, the estimated parameters of a particular type–action dynamics model are close to the true parameters:

$$T = \max\{T_\beta, T_\sigma\} = O\left(\frac{N_{dim}B^4}{\epsilon^2}\ln\frac{N_{dim}}{\delta}\right).$$

4.2 PLANNING ERROR

We next bound the error between the value function found by solving our particular continuous-state Markov decision process using fitted value iteration compared to the optimal value function V^*. Recall that by performing FVI, we are essentially mapping the original MDP to a new finite-state MDP where the states are the chosen fixed points.

Under a set of four assumptions, Chow and Tsitsiklis (1991) proved that the optimal value function V_ε of a discrete-state MDP formed by discretizing a continuous-state MDP into O(ε)-length (per dimension)[4] grid cells is an ε-close approximation of the optimal continuous-state MDP value function V^*:

$$||V_\varepsilon - V^*|| \leq \varepsilon.$$

The first two assumptions used to prove the above result are that the reward function and probability distribution are Lipschitz-continuous. In our work, the reward function is assumed to be given so this condition is a prior condition on the problem specification. Our probability distributions are Gaussian distributions that are Lipschitz-continuous so the second condition holds. The third key assumption is that the dynamics probabilities represent a true probablity measure that sums to 1 ($\int_s' p(s'|s,a) = 1$), though the authors show that this assumption can be relaxed to $\int_s' p(s'|s,a) \leq 1$ and the main results still hold. In our work, our dynamics models are defined to be true probability models. Chow and Tsitsiklis's final assumption is that there is a bounded difference between any two controls: in our case we consider only finite controls, so this property holds directly.

In summary, assuming the reward model fulfills the first assumption, our framework satisfies all four assumptions made by Chow and Tsitsiklis. Therefore, by selecting fixed grid points at a regular spacing of O(ϵ_{FVI}) in each dimension (letting $\varepsilon = \epsilon_{FVI}$), we can ensure that $||\tilde{V}_{FVI} - V^*||_\infty$ is at most ϵ_{FVI} where $\tilde{V}_{FVI}$ is the FVI optimal value function.

[4] More specifically, the grid spacing h_g must satisfy $h_g \leq \frac{(1-\gamma)^2\varepsilon}{K_1+2KK_2}$ and $h_g \leq \frac{1}{2K}$ where K is the larger of the Lipschitz constants arising from the assumptions discussed in the text, and K_1 and K_2 are constants discussed in Chow and Tsitsiklis (1991). For small ε any h_g satisfying the first condition will automatically satisfy the second condition.

4.3 APPROXIMATE REINFORCEMENT LEARNING

The next lemma relates the accuracy in the dynamics model parameters, and the error induced by approximate planning, to the value function of two MDPs. The proof strongly parallels a similar Simulation Lemma in recent work by Strehl and Littman (2008).

Lemma 4.5 *Let $M_1 = \langle S, A, p_1(s'|s,a), R, \gamma \rangle$ and $M_2 = \langle S, A, p_2(s'|s,a), R, \gamma \rangle$ be two MDPs[5] with dynamics as characterized in Equation 1 and non-negative rewards bounded above by 1. Assume $\frac{||\beta_1 - \beta_2||_2}{\sqrt{2\pi}\sigma_{\min}} \leq F_1$ and $|1 - \frac{|\Sigma_1|^{0.5}}{|\Sigma_2|^{0.5}}| \leq F_2$. Also assume that the difference between the value function $\tilde{V}$ obtained by fitted value iteration (FVI) compared to the optimal value function V^*, $||\tilde{V} - V^*||_\infty$ is at most F_3. Let π be a policy that can be applied to both M_1 and M_2. Then, for any $0 < \epsilon \leq V_{\max}$ and stationary policy π, if $F_1 = O(\frac{(1-\gamma)^2\epsilon}{\gamma})$ $F_2 = O(\frac{\epsilon(1-\gamma)^2}{\gamma})$, and $F_3 = O(\frac{\epsilon(1-\gamma)}{\gamma})$, then for all states s and actions a, $|Q_1^\pi(s,a) - \tilde{Q}_2^\pi(s,a)| \leq \epsilon$,where $\tilde{Q}_2^\pi$ denotes the state–action value obtained by performing FVI on MDP M_2 and Q_1^π denotes the true state–action value for MDP M_1 for policy π.*

Proof : (Sketch) We analyze the norm between $Q_1^\pi(s,a)$ and $\tilde{Q}_2^\pi(s,a)$ by re-expressing each in terms of its respective Bellman operator. The main idea is to break the norm up into a difference between the values due to the different dynamics ($p_1(s'|s,a)$ and $p_2(s'|s,a)$) and a difference due to using fitted value iteration to approximately solve for the values versus an exact solution. We use the triangle inequality to separate these terms and then bound each term separately, using the results from the prior sections. □

4.4 APPROXIMATELY OPTIMAL REINFORCEMENT LEARNING

Theorem 4.6 *For any given δ and ϵ in a continuous-state noisy offset dynamics MDP with N_T types where the variance along each dimension of all the dynamics models is bounded by $[\sigma_{\min}^2, B_\sigma^2]$ and the offset parameter is bounded by $|\beta_i| < B_\beta$ on all but N_{total} timesteps, our algorithm will follow a 4ϵ-optimal policy from its current state with probability at least $1-2\delta$, where N_{total} is polynomial in the problem parameters $(N_{dim}, |A|, N_T, \frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{1-\gamma}, \frac{1}{\sigma_{\min}}, B_\beta, B_\sigma)$.*

Proof : We demonstrate that our algorithm fulfills the three criteria outlined earlier. We omit details due to space considerations, but it can be shown using the results of the prior sections that after $N_{at} = O\left(\frac{N_{dim}^3 B^4 \gamma^2}{\sigma_{\min}^4 (1-\gamma)^4 \epsilon^2}\right)$ samples, with probability $1-\delta$, the errors $\|\beta_1 - \beta_2\|_2$, and for each state dimension i, $|\sigma_i^2 - \tilde{\sigma_i^2}|$ will be $O((1-\gamma)^2\epsilon)$. We also chose the spacing of our fixed grid points such that $\frac{\epsilon_{FVI}\gamma}{1-\gamma} \leq \frac{\epsilon}{2}$. Then, the Simulation Lemma (4.5) guarantees that the approximate value of our known state MDP solved using FVI is ϵ-close to the optimal value of the known state MDP with the true dynamics parameters $||\tilde{V}_{\tilde{K}}^\pi - V_K^\pi||_\infty \leq \epsilon$. All unknown type–action pairs that have not yet been experienced N_M times are considered to be unknown and their value is set to $V_{\max}$. So, Conditions (1) and (2) (Strehl et al., 2006) hold. The third condition limits the number of times the algorithm may experience an unknown type–action tuple. Since there are a finite number of types and actions, this quantity is bounded above by $N_{at} N_T |A|$, which is a polynomial in the problem parameters $(N_{dim}, |A|, N_T, \frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{1-\gamma}, \frac{1}{\sigma_{\min}}, B_\beta, B_\sigma)$. Therefore, our algorithm fulfills the three criteria laid out and the result follows. □

[5]For simplicity we present the results here without reference to types. In practice, each dynamics parameter would be subscripted by its associated MDP, type, and action.

5 EXPERIMENT

To examine the performance of our algorithm, we performed experiments in a real-life robotic environment involving a navigation task where a robotic car must traverse multiple surface types to reach a goal location. Our experiments seek to demonstrate both that our dynamics models provide a sufficiently good representation of real-world dynamics to allow our algorithm to learn good policies, and that our algorithm is computationally tenable. We demonstrate the second quality by comparing to Leffler et al. (2007)'s RAM-Rmax algorithm, a provably efficient RL algorithm for learning in discrete-state worlds with types. The authors demonstrated that, by explicitly representing the types, they could get a significant learning speedup compared to Rmax, which learns a separate dynamics model for each state. The RAM-Rmax algorithm represents the dynamics model using a list of possible next outcomes for a given type. Our approach instead assumes a fixed parametric distribution that automatically constrains the size of the representation.

5.1 EXPERIMENTAL SETUP

For our experiment, we ran a LEGO® Mindstorms NXT robot on a multi-surface environment. A tracking pattern was placed on the top of the robot and an overhead camera was used to determine the robot's current position and orientation. The domain, shown in Figure 2, consisted of two types: rocks embedded in wax and a carpeted area. The goal was for the agent to begin in the start location (indicated in the figure by an arrow) and end in the goal without going outside the environmental boundaries. The rewards were -1 for going out of bounds, $+1$ for reaching the goal, and -0.01 for taking an action. Reaching the goal and go-

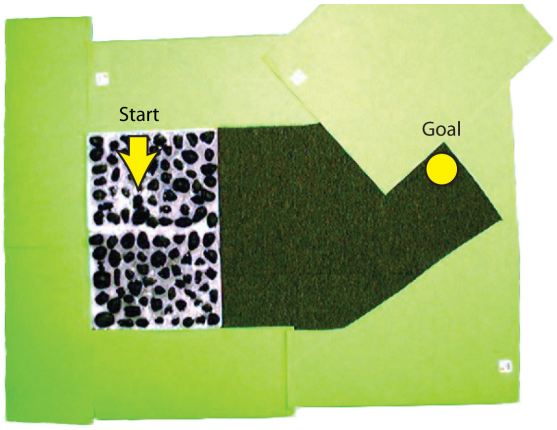


Figure 2: Image of the environment. The start location and orientation is marked with an arrow. The goal location is indicated by the circle.

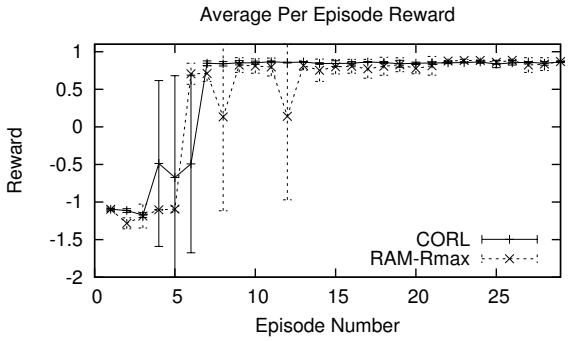


Figure 3: Reward received by algorithms averaged over three runs. Error bars show one standard deviation.

ing out of bounds ended the episode and resulted in the agent getting moved back to the start location.

One difficulty of this environment is the difference in dynamics models. Due to the close proximity of the goal to the boundary, the agent needs an accurate dynamics model to reliably reach the goal. To make this task even more difficult, the actions were limited to going forward, turning left, and turning right. Without the ability to move backwards, the robot needed to approach the goal accurately to avoid falling out of bounds. A robot with an inaccurate transition model would be likely to judge this task as impossible.

For the experiments, we compared our algorithm ("CORL") and the RAM-Rmax algorithm ("RAM"). The fixed points for the fitted value iteration portion of our algorithm were set to the discretized points of the RAM-Rmax algorithm. Both algorithms used an EDISON image segmentation system to uniquely identify the current surface type. The reward function was provided to both algorithms.

The state space is three dimensional: x, y, and orientation. Our algorithm implementation for this domain used a full covariance matrix to model the dynamic's variance model. For the RAM-Rmax agent, the world was discretized to a forty-by-thirty-by-ten state space. In our algorithm, we used a function approximator of a weighted sum of Gaussians, as described in Section 2.2. We used the same number of Gaussians to represent the value function as the size of the state space used in the discretized algorithm, and placed these fixed Gaussians at the same locations. The variance over the x and y variables was independent of each other and of orientation, and was set to be 16. To average orientation vectors correctly (so that -180° degrees and 180° do not average to 0) we converted orientations θ to a Cartesian coordinate representation $x_\theta = \cos(\theta), y_\theta = \sin(\theta)$. The variance over these two was set to be 9 for each variable (with zero covariance). For our algorithm and the RAM-Rmax algorithm, the value of N_{at} was set to four and five, respectively, which was determined after informal experimentation. The discount factor was set to 1.

5.2 RESULTS

Figure 3 shows the average reward with standard deviation for each of the algorithms over three runs. Both algorithms are able to receive near-optimal reward on a consistent basis choosing similar paths to the goal. Our dynamics representation is sufficient to allow our algorithm to learn well in this real-life environment.

Examining the learned dynamics model parameters revealed that the dynamics model variances learned for the rocks were larger than the learned variance for carpet for certain actions. Naturally, an important question is whether modeling the differences in the dynamics models is necessary in order to achieve good performance: in other words, could the robot perform as well by modeling the terrain as a single type? Prior work by RAM-Rmax on a similar task compared using two types to one, and found that two types did result in a better learned policy(Leffler et al., 2008). This finding suggests that using multiple types to represent this environment provides measurable benefits.

In addition, by using a fixed parametric representation, the computational time per episode of our algorithm is roughly constant. In the implementation of RAM-Rmax, the computational time grew with the number of episodes due to its dynamics model representation: however, this difficulty could be ameliorated by maintaining a finite list of potential dynamics transitions. Nonetheless, these results suggest that our algorithm is computationally competitive with existing approaches to handle domains with typed dynamics.

In summary, the results on this task are encouraging since they indicate our algorithm can quickly and efficiently learn

a good policy in a real-world environment with switching noisy offset dynamics.

6 CONCLUSION

We have presented a new reinforcement-learning algorithm for handling continuous-state typed worlds where the dynamics can be modeled as a noisy offset. In this work, we have assumed that the state types are fully observable. This assumption is likely to be realistic for certain domains, such as when types correspond to the slope of an outdoor environment in which contour maps are available. In other cases, it might be useful to model the type as a hidden variable, and receive estimates of it through the agent's sensors. Such a scenario is beyond the scope of this paper but would be interesting future work.

In conclusion, we proved that when the noise covariance matrix is diagonal, the algorithm is probably approximately correct with a sample complexity that scales polynomially with the MDP parameters, including the state-space dimension. We also demonstrated that in some scenarios these dynamics representations can provide a sufficiently good approximation of real-world dynamics to enable a good policy to be learned by demonstrating the success of our algorithm in a small robotic experiment.

Acknowledgements

B. Leffler, L. Li and M. Littman were partially supported by NSF DGE 0549115, a DARPA transfer learning grant and a National Science Foundation (NSF) Division of Information and Intelligent Systems (IIS) grant. E. Brunskill and N. Roy were supported by NSF IIS under Grant #0546467.

References

Abbeel, P., & Ng, A. Y. (2005). Exploration and apprenticeship learning in reinforcement learning. *Proceedings of the 22nd International Conference on Machine Learning* (pp. 1–8).

Brafman, R. I., & Tennenholtz, M. (2002). R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, *3*, 213–231.

Brunskill, E., Leffler. B. R., Li, L., Littman, M. L., & Roy, N. (2008). *CORL: A continuous-state offset-dynamics reinforcement learner.* Technical Report. Massachusetts Institute of Technology.

Chow, C., & Tsitsiklis, J. (1991). An optimal multigrid algorithm for continuous state discrete time stochastic control. *IEEE Transactions on Automatic Control*, *36*, 898–914.

Gordon, G. (1995). Stable function approximation in dynamic programming. *Proceedings of the 12th International Conference on Machine Learning* (pp. 261–268).

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*, 13–30.

Kakade, S. (2003). *On the sample complexity of reinforcement learning*. Doctoral dissertation, University College London.

Kearns, M. J., & Singh, S. P. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, *49*, 209–232.

Leffler, B. R., Littman, M. L., & Edmunds, T. (2007). Efficient reinforcement learning with relocatable action models. *AAAI-07: Proceedings of the Twenty-Second Conference on Artificial Intelligence* (pp. 572–577). Menlo Park, CA, USA: The AAAI Press.

Leffler, B. R., Mansley, C. R., & Littman, M. L. (2008). *Efficient learning of dynamics models using terrain classification.* Technical Report DCS-tr-633. Rutgers University.

Ng, A., Kim, H., Jordan, M., & Sastry, S. (2004). Autonomous helicopter flight via reinforcement learning. *Neural Information Processing Systems 16*.

Strehl, A., Li, L., & Littman, M. (2006). Incremental model-based learners with formal learning-time guarantees. *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*.

Strehl, A., & Littman, M. (2008). Online linear regression and its application to model-based reinforcement learning. *Neural Information Processing Systems 20*.

Tesauro, G. J. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, *6*, 215–219.

On Identifying Total Effects in the Presence of Latent Variables and Selection bias

Zhihong Cai
Department of Biostatistics
School of Public Health
Kyoto University
cai@pbh.med.kyoto-u.ac.jp

Manabu Kuroki
Department of Systems Innovation
School of Engineering Science
Osaka University
mkuroki@sigmath.es.osaka-u.ac.jp

Abstract

Assume that cause-effect relationships between variables can be described as a directed acyclic graph and the corresponding linear structural equation model. We consider the identification problem of total effects in the presence of latent variables and selection bias between a treatment variable and a response variable. Pearl and his colleagues provided the back door criterion, the front door criterion (Pearl, 2000) and the conditional instrumental variable method (Brito and Pearl, 2002) as identifiability criteria for total effects in the presence of latent variables, but not in the presence of selection bias. In order to solve this problem, we propose new graphical identifiability criteria for total effects based on the identifiable factor models. The results of this paper are useful to identify total effects in observational studies and provide a new viewpoint to the identification conditions of factor models.

1 INTRODUCTION

The evaluation of total effects from observational studies is one of the central aims in many fields of practical science. In observational studies, there may exist latent variables, for example, a variable measured with error, or an unmeasured confounder. On the other hand, observational data may suffer from selection bias, if a sample is selected according to some selection criteria. The existence of latent variables and selection bias hinder the evaluation of total effects from observational data. Many researchers have provided approaches to deal with latent variables in observational studies (Brito and Pearl, 2002; Pearl, 2000; Stanghellini, 2004; Stanghellini and Wermuth, 2005; Tian, 2004). Recently, selection bias has attracted attention from epidemiologists (Greenland, 2003; Hernan et al., 2004), AI researchers (Cooper, 2000) and statisticians (Stanghellini and Wermuth, 2005; Kuroki and Cai, 2006).

In observational studies, it is not rare that both latent variables and selection bias exist in one dataset. However, when we examine current results on latent variables and selection bias, we find that most of them deal with either latent variables or selection bias separately, only a few of them take into account these two situations at the same time. In the presence of latent common causes between a treatment and a response, Pearl and his colleagues provided the back door criterion, the front door criterion (Pearl, 2000) and the conditional instrumental variable (IV) method (Brito and Pearl, 2002) as identifiability criteria for total effects in the framework of linear structural equation models. In addition, in the framework of nonparametric structural equation models, Shpitser and Pearl (2006) and Huang and Valtorta (2006) solved the identification problems of causal effects and provided the complete algorithms to derive the causal effects in the presence of latent variables. In general, these criteria are based on the idea that some observed variables which have no direct association with the latent common causes are used to evaluate the total effects. However, in many practical studies, such latent variables may have an effect on some important observed variables to be used to identify total effects. Under such situations, it is difficult to apply these identification criteria to evaluate the total effects.

On the other hand, when both confounding bias and selection bias may be at work, Spirtes et al. (1999) described the FCI algorithm (Spirtes et al., 2000) as a method to test whether there is a causal path from one variable to another. In addition, Richardson and Spirtes (2002) introduced ancestral graph models as a graphical model in the presence of latent variables and selection bias, and clarified some properties regarding the ancestral graph models. However, these studies fo-

cused on the specification problem of causal structure, but not on the identification problem of total effects.

In this paper, we assume that cause-effect relationships between variables can be described as a directed acyclic graph and the corresponding linear structural equation model. Then, we consider the problem of identifying total effects from observational studies with latent variables and selection bias between a treatment variable and a response variable. Based on the theory of the identifiable factor model, we propose new graphical identifiability criteria to identify total effects under situations where it is difficult to use the identifiability criteria provided by Pearl and his colleagues and Huang and Valtorta (2006). Different from the identification problem of the factor models, it should be noted that we are interested in evaluating the total effects but not the whole causal model. That is, it will be shown in section 3 that there are some situations where the total effect is identifiable even when the whole causal model is not identifiable. These new criteria are useful to identify total effects in observational studies, and they also provide a new viewpoint to the identification conditions of factor models.

2 PRELIMINARIES

In statistical causal analysis, a directed acyclic graph that represents cause-effect relationships is called a path diagram. A directed graph is a pair $G = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ is a finite set of vertices and the set $\boldsymbol{E}$ of arrows is a subset of the set $\boldsymbol{V} \times \boldsymbol{V}$ of ordered pairs of distinct vertices. For graph theoretic terminology used in this paper, see, for example, Lauritzen (1996).

Suppose a directed acyclic graph $G = (\boldsymbol{V}, \boldsymbol{E})$ with a set $\boldsymbol{V} = \{V_1, V_2, \cdots, V_n\}$ of variables is given. The graph G is called a path diagram, when each child-parent family in the graph G represents a linear structural equation model

$$V_i = \sum_{V_j \in \mathrm{pa}(V_i)} \alpha_{v_i v_j} V_j + \epsilon_{v_i} \qquad i = 1, \ldots, n, \quad (1)$$

where pa (V_i) is a set of parents of V_i. In this paper, if there is no special statement, $\epsilon_{v_1}, \ldots, \epsilon_{v_n}$ are assumed to be independent and normally distributed with mean 0. In addition, $\alpha_{v_i v_j} (\neq 0)$ is called a path coefficient.

The conditional independence induced from a set of equations (1) can be obtained from the graph G according to the d-separation (Pearl, 2000), that is, when $\boldsymbol{Z}$ d-separates X from Y in a path diagram G, X is conditionally independent of Y given $\boldsymbol{Z}$ in the corresponding linear structural equation model (e.g. Spirtes et al., 2000). In this paper, it is assumed that a path diagram G and the corresponding joint distribution are faithful to each other; that is, the conditional independence relationship in the joint distribution is also reflected in G, and vice versa (Spirtes et al., 2000).

Here, we denote some notations for further discussion. Let $\sigma_{xy \cdot zs^*} = \mathrm{cov}(X, Y | \boldsymbol{Z} = \boldsymbol{z}, \boldsymbol{a} \leq \boldsymbol{S} \leq \boldsymbol{b})$ and $\sigma_{yy \cdot zs^*} = \mathrm{var}(Y | \boldsymbol{Z} = \boldsymbol{z}, \boldsymbol{a} \leq \boldsymbol{S} \leq \boldsymbol{b})$ and $\beta_{yx \cdot zs^*} = \sigma_{xy \cdot zs^*} / \sigma_{xx \cdot zs^*}$ ($\boldsymbol{s}^*$ indicates that each element of $\boldsymbol{s}$ is conditioned by the interval). For disjoint sets $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{S}$, let $\Sigma_{xy \cdot zs^*}$ be a conditional covariance matrix of $\boldsymbol{X}$ and $\boldsymbol{Y}$ given $\boldsymbol{Z} = \boldsymbol{z}$ and $\boldsymbol{a} \leq \boldsymbol{S} \leq \boldsymbol{b}$. We use the same notations in the case where either $\boldsymbol{X}$ or $\boldsymbol{Y}$ is univariate. In addition, let $\Sigma_{yy \cdot xs^*}$ be a conditional covariance matrix of $\boldsymbol{Y}$ given $\boldsymbol{X} = \boldsymbol{x}$ and $\boldsymbol{a} \leq \boldsymbol{S} \leq \boldsymbol{b}$. When $\boldsymbol{S}$ or $\boldsymbol{Z}$ is an empty set, they are omitted from these arguments. Furthermore, let $B_{yx \cdot z} = \Sigma_{yx \cdot z} \Sigma_{xx \cdot z}^{-1}$ be the regression coefficient matrix of $\boldsymbol{x}$ in the regression model of $\boldsymbol{Y}$ on $\boldsymbol{x} \cup \boldsymbol{z}$. The similar notations are used for other parameters.

A total effect τ_{yx} of X on Y is defined as the total sum of the products of the path coefficients on the sequence of arrows along all directed paths from X to Y. In this paper, it is assumed that the readers are familiar with the identifiability criteria for total effects, for example, the back door criterion, the front door criterion (Pearl, 2000) and the IV method (Bowden and Turkington, 1984; Brito and Pearl, 2000). When a total effect can be determined uniquely from the covariance parameters of observed variables, it is said to be identifiable, that is, it can be estimated consistently.

When $\boldsymbol{Z}$ d-separates X from Y in a path diagram G, then both $\sigma_{xy \cdot z} = \beta_{yx \cdot z} = 0$ and $B_{yz \cdot x} = B_{yz}$ hold true (e.g. Spirtes et al., 2000).

3 IDENTIFICATION OF TOTAL EFFECTS

3.1 LEMMA

To derive new graphical identifiability criteria for total effects, we first introduce the following lemmas:

LEMMA 1

When $\{X, Y\} \cup \boldsymbol{S} \cup \boldsymbol{T}$ are normally distributed,

$$\beta_{yx \cdot s} = \beta_{yx \cdot st} + B_{yt \cdot xs} B_{tx \cdot s}, \quad (2)$$

$$\sigma_{yy \cdot xs} = \sigma_{yy \cdot x} - B_{ys \cdot x} \Sigma_{ss \cdot x} B'_{ys \cdot x}. \quad (3)$$

□

Equations (2) and (3) are the results of Cochran (1938) and Whittaker (1990), respectively. In addition, the following lemma is given by Wermuth (1989).

LEMMA 2

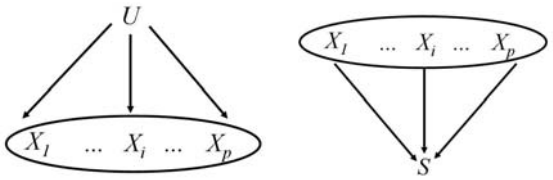
When $\{X, Y\} \cup \boldsymbol{S} \cup \boldsymbol{T}$ are normally distributed, if $\boldsymbol{T}$ is conditionally independent of X given $\boldsymbol{S}$ or Y is condi-

tionally independent of $\boldsymbol{T}$ given $\{X\}\cup\boldsymbol{S}$, then $\beta_{yx\cdot st} = \beta_{yx\cdot s}$ holds true. In addition, if $\boldsymbol{T}$ is conditionally independent of Y given $\boldsymbol{S}\cup\{X\}$, then $\sigma_{yy\cdot xst} = \sigma_{yy\cdot xs}$ holds true. □

3.2 DUALITY BETWEEN LATENT VARIABLES AND SELECTION BIAS

In this section, we consider two different situations: one is a situation where a latent variable exists shown in Fig.1 (a); the other is a situation where selection bias exists shown in Fig.1 (b), where $\boldsymbol{X} = (X_1, \cdots, X_p)$ is a set of observed variables, and U is a latent variable which has an effect on $\boldsymbol{X}$. In addition, Fig.1(b) indicates that the data have been observed according to the selection criterion $a \leq S \leq b$ (both a and b are possible values of S).

Regarding Fig.1 (a), the corresponding linear structural equation model can be provided as



(a): Latent Variable Case (b): Selection Bias Case

Fig.1: Graphical representation

$$X_i = \alpha_{x_i u} U + \epsilon_{x_i}, \qquad i = 1, \ldots, p, \tag{4}$$

which is called a single factor model (with correlated errors). Then, the covariance matrix of $\boldsymbol{X}$ can be provided as

$$\begin{aligned} \Sigma_{xx} &= \Sigma_{xx\cdot u} + B_{xu} B'_{xu} \sigma_{uu} \\ &= \Sigma_{xx\cdot u} + \Omega(u)\Omega(u)', \end{aligned} \tag{5}$$

where $\Omega(u)$ is a p dimensional vector, which is called a factor loading in this paper. Here, it should be noted that Σ_{xx} can be observed but σ_{uu}, $\Sigma_{xx\cdot u}$ or Σ_{xu} can not be observed.

On the other hand, regarding the Σ_{xx}^{-1}, by the Sherman-Morrison-Woodbury formula for matrix inversion (Rao, 1972), the inverse matrix of Σ_{xx} can be represented as the form of

$$\Sigma_{xx}^{-1} = \Sigma_{xx\cdot u}^{-1} - \Lambda(u)\Lambda(u)', \tag{6}$$

where $\Lambda(u)$ is a p dimensional vector.

Regarding Fig.1 (b), the corresponding linear structural equation model can be provided as

$$S = \sum_{j=1}^{p} \alpha_{s_i x_j} X_j + \epsilon_{s_i}. \tag{7}$$

Then, the covariance matrix of the selected population can be provided as

$$\begin{aligned} \Sigma_{xx\cdot s^*} &= \Sigma_{xx} - B_{xs} B'_{xs} (\sigma_{ss} - \sigma_{s^* s^*}) \\ &= \Sigma_{xx} - \Omega(s)\Omega(s)', \end{aligned} \tag{8}$$

where $\Omega(s)$ is a p dimensional vector (Johnson and Kotz, 1972). Here, $\check{\sigma}_{ss} = \sigma_{ss} - \sigma_{ss\cdot s^*} \geq 0$ since $\sigma_{ss\cdot s^*} = \mathrm{var}(S | a \leq S < \leq b)$ is the variance of a doubly-truncated normal distribution. In the selected population, it should be noted that $\Sigma_{xx\cdot s^*}$ can be observed but Σ_{xx}, $\check{\sigma}_{ss}$ or B_{xs} can not be observed.

On the other hand, regarding $\Sigma_{xx\cdot s^*}^{-1}$, we can obtain

$$\Sigma_{xx\cdot s}^{-1} = \Sigma_{xx}^{-1} + \Lambda(s)\Lambda(s)', \tag{9}$$

where $\Lambda(s)$ is a p dimensional vector, which is also called a factor loading in this paper.

In addition, when we discuss latent variable problems and selection bias problems based on conditional distribution given $\boldsymbol{Z}$, the following equations hold true:

$$\Sigma_{xx\cdot z} = \Sigma_{xx\cdot uz} + B_{xu\cdot z} B'_{xu\cdot z} \sigma_{uu\cdot z}$$

and

$$\Sigma_{xx\cdot zs^*} = \Sigma_{xx\cdot z} - B_{xs\cdot z} B'_{xs\cdot z} \check{\sigma}_{ss\cdot z},$$

where $\check{\sigma}_{ss\cdot z} = \sigma_{ss\cdot z} - \sigma_{ss\cdot zs^*} \geq 0$.

From these equations, we can understand that equations (5) and (6) take the same form as equations (9) and (8), respectively. In this paper, such relationships are called the duality between latent variables and selection bias. By using the dual relationships, we will show below that the identification conditions of factor models are useful to solve the selection bias problems.

Let $G_{cov}^{x\cdot u}$ be the undirected graph obtained by connecting any two variables X_i and X_j $(i \neq j)$ in $\boldsymbol{X}$ by an undirected edge only if the conditional covariance of X_i and X_j given U is not equal to zero. Let $G_{con}^{x\cdot u}$ be the undirected graph obtained by connecting any two variables X_i and X_j $(i \neq j)$ in $\boldsymbol{X}$ by an undirected edge only if the conditional covariance of X_i and X_j given $\{U\} \cup \boldsymbol{X} \backslash \{X_i, X_j\}$ is not equal to zero. When we are concern with the covariance structure of $\boldsymbol{X}$ not conditioning on U, U are omitted from these arguments.

Then, Stanghellini and Wermuth (2005) provided the following lemma.

LEMMA 3

Equations (5)(or equation (6)) can be solved with respect to $\Sigma_{xx\cdot u}$ and $\Omega(u)\Omega(u)'$ (or $\Sigma_{xx\cdot u}^{-1}$ and $\Lambda(u)\Lambda(u)'$) if and only if one of the following conditions holds true:

(1) $\Omega(u) \neq \mathbf{0}$ and the structure of zeros in $\Sigma_{xx\cdot u}$ is such that every connectivity component of the complementary graph of $G_{cov}^{x\cdot u}$ contains an odd cycle;

(2) $\Lambda(u) \neq \mathbf{0}$ and the structure of zeros in $\Sigma_{xx \cdot u}^{-1}$ is such that every connectivity component of the complementary graph of $G_{con}^{x \cdot u}$ contains an odd cycle. □

The similar results hold true for equations (8) and (9)

3.3 IDENTIFIABILITY CRITERION: LATENT VARIABLE CASE

It is well known that the graphical identifiability criteria proposed by Pearl and his colleagues are useful to evaluate total effects. However, Stanghellini (2004) pointed out that there are some situations where these identifiability criteria can not be applied to evaluate total effects. As an example, we consider the problem of evaluating the total effect τ_{yx} of X on Y based on the path diagram shown in Fig. 2, where U is an unobserved variable and $\{X, Y, Z, W\}$ is a set of observed variables.

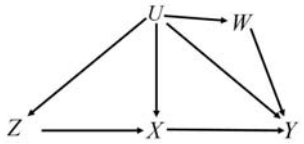


Fig. 2: Path Diagram (1)

In Fig. 2, since U is an unobserved variable, we can not apply the back door criterion relative to (X, Y) to evaluate the total effect τ_{yx}. In addition, since we can not observe a set of variables that satisfies the front door criterion relative to (X, Y), the front door criterion can not be applied, either. Furthermore, since there are arrows pointing from the unobserved variable U to every observed variable, the conditional IV method can not be applied. Under such a situation, it is necessary to propose new identifiability criteria different from current results.

When we consider the linear structural equation model corresponding to the directed acyclic graph obtained by deleting from Fig. 2 an arrow pointing from W to Y (i.e., $\alpha_{yw} = 0$), since we can obtain the same covariance structure as the identifiable single factor model (e.g. Stanghellini, 1997), the total effect τ_{yx} is identifiable (Stanghellini, 2004). However, in Fig. 2, since there is an arrow pointing from W to Y, the number of observed covariances is less than that of the path coefficients, which indicates that the whole linear structural equation model can not be identifiable even if the variance information on U is known (e.g. $\sigma_{uu} = 1$). However, the path coefficient α_{yx} is identifiable. This result is summarized as follows (Kuroki, 2007):

THEOREM 1

Suppose that a set $\{X, Y, W, Z\} \cup \boldsymbol{T}$ of observed variables and an unobserved variable U satisfy the following conditions in a directed acyclic graph G:

(1) $\{X, U\} \cup \boldsymbol{T}$ d-separates Y from Z,

(2) $\{U\} \cup \boldsymbol{T}$ d-separates $\{X, Z\}$ from W, and

(3) $\{X\} \cup \boldsymbol{T}$ does not d-separate Z from W.

When X is an nondescendant of Y, if $\{U\} \cup \boldsymbol{T}$ satisfies the back door criterion relative to (X, Y), then the total effect τ_{yx} of X on Y is identifiable and is given by the formula

$$\tau_{yx} = \frac{\sigma_{xw \cdot t}\sigma_{yz \cdot t} - \sigma_{zw \cdot t}\sigma_{xy \cdot t}}{\sigma_{xw \cdot t}\sigma_{zx \cdot t} - \sigma_{zw \cdot t}\sigma_{xx \cdot t}}. \tag{10}$$

□

PROOF OF THEOREM 1

Since $\{U\} \cup \boldsymbol{T}$ satisfies the back door criterion relative to (X, Y), $\tau_{yx} = \beta_{yx \cdot ut}$ can be obtained. In addition, from Lemma 1, the following can be derived:

$$\begin{aligned}
\sigma_{yz \cdot t} &= \beta_{yz \cdot xut}\sigma_{zz \cdot t} + \beta_{yx \cdot uzt}\sigma_{xz \cdot t} + \beta_{yu \cdot xzt}\sigma_{uz \cdot t}, \\
\sigma_{xy \cdot t} &= \beta_{yx \cdot ut}\sigma_{xx \cdot t} + \beta_{yu \cdot xt}\sigma_{ux \cdot t}, \\
\sigma_{xz \cdot t} &= \beta_{zx \cdot tw}\sigma_{xx \cdot t} + \beta_{zw \cdot tx}\sigma_{wx \cdot t}, \\
\sigma_{zw \cdot t} &= \beta_{zw \cdot xt}\sigma_{ww \cdot t} + \beta_{zx \cdot tw}\sigma_{xw \cdot t}.
\end{aligned}$$

From condition (1), since Y is conditionally independent of Z given $\{X, U\} \cup \boldsymbol{T}$, $\beta_{yz \cdot xut} = 0$ can be obtained. In addition, by using Lemma 2, we can obtain $\beta_{yx \cdot uzt} = \beta_{yx \cdot ut}$ and $\beta_{yu \cdot xzt} = \beta_{yu \cdot xt}$. Noting these results, we have

$$\sigma_{yz \cdot t} = \beta_{yx \cdot ut}\sigma_{xz \cdot t} + \beta_{yu \cdot xt}\sigma_{uz \cdot t}.$$

Then, we can obtain

$$\begin{aligned}
&\sigma_{xw \cdot t}\sigma_{yz \cdot t} - \sigma_{zw \cdot t}\sigma_{xy \cdot t} \\
&= \beta_{yx \cdot ut}(\sigma_{xw \cdot t}\sigma_{xz \cdot t} - \sigma_{zw \cdot t}\sigma_{xx \cdot t}) \\
&\quad + \beta_{yu \cdot xt}(\sigma_{xw \cdot t}\sigma_{uz \cdot t} - \sigma_{zw \cdot t}\sigma_{ux \cdot t}).
\end{aligned}$$

Here, since $\beta_{zw \cdot tx} \neq 0$ holds true from condition (3) and the faithful condition, from Lemma 1 and

$$\begin{aligned}
\sigma_{xw \cdot t} &= \beta_{xw \cdot ut}\sigma_{ww \cdot t} + \beta_{xu \cdot tw}\sigma_{uw \cdot t}, \\
\sigma_{zw \cdot t} &= \beta_{zw \cdot ut}\sigma_{ww \cdot t} + \beta_{zu \cdot tw}\sigma_{uw \cdot t},
\end{aligned}$$

we can obtain

$$\begin{aligned}
&\sigma_{xw \cdot t}\sigma_{xz \cdot t} - \sigma_{zw \cdot t}\sigma_{xx \cdot t} \\
&= -\beta_{zw \cdot tx}\sigma_{ww \cdot t}(\sigma_{xx \cdot t} - \beta_{xw \cdot t}\sigma_{wx \cdot t}) \\
&= -\beta_{zw \cdot tx}\sigma_{ww \cdot t}\sigma_{xx \cdot tw} \neq 0.
\end{aligned}$$

Thus, by noting that $\beta_{xw \cdot ut} = \beta_{zw \cdot ut} = 0$ can be obtained from condition (2), we have

$$\sigma_{xw \cdot t}\sigma_{uz \cdot t} - \sigma_{zw \cdot t}\sigma_{ux \cdot t} = 0.$$

By noting these results, equation (10) can be derived. *Q.E.D.*

It should be noted that the assumption of the variance of an unobserved variable U is not required in Theorem 1, which is different from the identification condition of factor models (e.g. Stanghellini, 1997).

In the case where there are more than one unmeasured confounder, Kuroki (2007) pointed out that the identification condition for multi-factor models (e.g.Grzebyk et al., 2004) is also useful to identify the total effects. The results can be summarized as follows.

THEOREM 2

Let $\boldsymbol{X} = \{X_1, \cdots, X_p\}$ be a set of observed variables and $\boldsymbol{U} = \{U_1, \cdots, U_k\}$ a set of unobserved variables in the path diagram G. When a linear structural equation model obtained by conditioning on $\{U_1, \cdots, U_{i-1}\}$ and marginalizing on $\{U_{i+1}, \cdots, U_k\}$ is regarded as a single factor model of U_i, if the single factor model of U_i is identifiable for any $i(1 \leq i \leq k)$, $\Sigma_{xx \cdot u}$ is also identifiable.

PROOF OF THEOREM 2

First, the covariance matrix corresponding to the linear structural equation model which is marginalized on $\{U_2, \cdots, U_k\}$ is given by

$$\Sigma_{xx} = \Sigma_{xx \cdot u_1} + \frac{1}{\sigma_{u_1 u_1}} \Sigma_{x u_1} \Sigma'_{x u_1}.$$

Then, $\Sigma_{xx \cdot u_1}$ is identifiable from the assumption.

Here, we assume that $\Sigma_{xx \cdot u_1 \cdots u_{i-1}}$ is identifiable for $i(\geq 2)$. Then, the covariance matrix corresponding to the linear structural equation model which is marginalized on $\{U_{i+1}, \cdots, U_k\}$ and conditioned on $\{U_1, \cdots, U_{i-1}\}$ is given by

$$\Sigma_{xx \cdot u_1 \cdots u_{i-1}}$$
$$= \Sigma_{xx \cdot u_1 \cdots u_i} + \frac{1}{\sigma_{u_i u_i \cdot u_1 \cdots u_{i-1}}} \Sigma_{x u_i \cdot u_1 \cdots u_{i-1}} \Sigma'_{x u_i \cdot u_1 \cdots u_{i-1}}.$$

Thus, $\Sigma_{xx \cdot u_1 \cdots u_i}$ is identifiable from the assumption. By repeating this procedure, the result can be obtained. *QED*

When Theorem 2 holds true for $\{X, Y\} \cup \boldsymbol{Z} \subset \boldsymbol{X}$, if $\boldsymbol{Z} \cup \boldsymbol{U}$ satisfies the back door criterion relative to (X, Y), the total effect τ_{yx} is identifiable.

As an example, we consider the problem of evaluating total effect τ_{yx} in the path diagram shown in Fig. 3. Although we can not apply the identifiability criteria proposed by Pearl and his colleagues, since Theorem 2 holds true, the total effect τ_{yx} is identifiable and is given by the formula

$$\tau_{yx} = \frac{\sigma_{yz_1} \sigma_{z_2 w_1} - \sigma_{y w_1} \sigma_{z_1 z_2}}{\sigma_{x z_1} \sigma_{z_2 w_1} - \sigma_{x w_1} \sigma_{z_1 z_2}}.$$

It is interesting that the above equation does not include the covariance parameter about W_2.

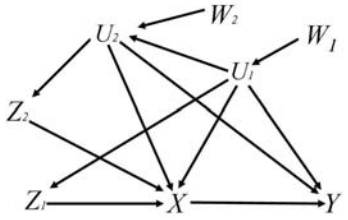


Fig. 3: Path Diagram (2)

3.4 IDENTIFAIBLITY CRITERION: SELECTION BIAS CASE

Selection bias is another case that the identifiability criteria proposed by Pearl and his colleagues can not be applied to evaluate total effects. Consider the identification problem for the total effect τ_{yx} based on the path diagram shown in Fig. 4, which indicates that sample selection is conducted according to a criterion $a \leq S \leq b$. Then, S is called a selection variable. In addition, $\{X, Y, Z, W\}$ is a set of observed variables.

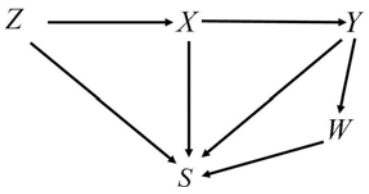


Fig. 4: Path Diagram (3)

In Fig. 4, since a sample is selected from the population using such a criterion as $a \leq S \leq b$, the statistical dependencies among $\{X, Y, Z, W\}$ are biased. Thus, we can not apply any identifiability criteria proposed by Pearl and his colleagues to identify the total effect. On the other hand, when we consider the linear structural equation model corresponding to the directed acyclic graph obtained by deleting from Fig. 4 an arrow pointing from Y to W(i.e. $\alpha_{wy} = 0$), since the number of the observed covariances is equal to that of the path coefficients, the total effect τ_{yx} can be evaluated through the observed covariances. However, in Fig. 4, since the number of the observed covariances is less than that of the path coefficients, the whole linear structural equation model is not identifiable. But, the total effect τ_{yx} is identifiable through the following theorem.

THEOREM 3

Suppose a set $\{X, Y, W, Z\} \cup \boldsymbol{T}$ of observed variables and a selection variable S satisfy the following conditions in a directed acyclic graph G:

(1) $\{X\} \cup \boldsymbol{T}$ d-separates Y from Z,

(2) $\boldsymbol{T}$ d-separates $\{X, Z\}$ from $\{W\}$,

(3) $\{X\} \cup \boldsymbol{T}$ does not d-separate S from Z, and

(4) $\boldsymbol{T}$ does not d-separate S from W.

When X is a nondescendant of Y, if $\boldsymbol{T}$ satisfies the back door criterion relative to (X, Y), then the total effect τ_{yx} of X on Y is identifiable and is given by the formula

$$\tau_{yx} = \frac{\sigma_{xw\cdot ts^*}\sigma_{yz\cdot ts^*} - \sigma_{zw\cdot ts^*}\sigma_{xy\cdot ts^*}}{\sigma_{xw\cdot ts^*}\sigma_{zx\cdot ts^*} - \sigma_{zw\cdot ts^*}\sigma_{xx\cdot ts^*}}. \quad (11)$$

□

PROOF OF THEOREM 3

Since $\boldsymbol{T}$ satisfies the back door criterion relative to (X, Y), $\tau_{yx} = \beta_{yx\cdot t}$ can be obtained. In addition, from Lemma 1 and condition (2),

$$\begin{aligned}
\sigma_{yz\cdot ts^*} &= \sigma_{yz\cdot xt} + \beta_{yx\cdot tz}\sigma_{xz\cdot t} - \beta_{ys\cdot t}\beta_{zs\cdot t}\check{\sigma}_{ss\cdot t} \\
\sigma_{xy\cdot ts^*} &= \sigma_{yx\cdot t} - \beta_{xs\cdot t}\beta_{ys\cdot t}\check{\sigma}_{ss\cdot t}, \\
\sigma_{zx\cdot ts^*} &= \sigma_{zx\cdot t} - \beta_{zs\cdot t}\beta_{xs\cdot t}\check{\sigma}_{ss\cdot t}, \\
\sigma_{zw\cdot ts^*} &= -\beta_{zs\cdot t}\beta_{ws\cdot t}\check{\sigma}_{ss\cdot t}, \\
\sigma_{xw\cdot ts^*} &= -\beta_{xs\cdot t}\beta_{ws\cdot t}\check{\sigma}_{ss\cdot t}.
\end{aligned}$$

From conditions (1), since Y is conditionally independent of Z given $\{X\}\cup\boldsymbol{T}$, $\sigma_{yz\cdot xt} = 0$ can be obtained. In addition, from Lemma 2, we can obtain $\beta_{yx\cdot zt} = \beta_{yx\cdot t}$. Using these results, we have

$$\sigma_{yz\cdot ts^*} = \beta_{yx\cdot t}\sigma_{xz\cdot t} - \beta_{ys\cdot t}\beta_{zs\cdot t}\check{\sigma}_{ss\cdot t}.$$

Thus, we can obtain

$$\begin{aligned}
&\sigma_{xw\cdot ts^*}\sigma_{yz\cdot ts^*} - \sigma_{zw\cdot ts^*}\sigma_{xy\cdot t}s^* \\
&= \beta_{yx\cdot t}\beta_{ws\cdot t}\check{\sigma}_{ss\cdot t}\sigma_{xx\cdot t}(\beta_{xs\cdot t}\beta_{zx\cdot t} - \beta_{zs\cdot t}) \\
&= -\beta_{yx\cdot t}\beta_{ws\cdot t}\check{\sigma}_{ss\cdot t}\sigma_{xx\cdot t}\sigma_{zs\cdot xt}/\sigma_{ss\cdot t}.
\end{aligned}$$

Here, since $\beta_{ws\cdot t}\beta_{zs\cdot xt}\neq 0$ holds true from conditions (3) and (4) and the faithful condition, according to Lemma 1, we can obtain

$$\begin{aligned}
&\sigma_{xw\cdot ts^*}\sigma_{xz\cdot ts^*} - \sigma_{zw\cdot ts^*}\sigma_{xx\cdot ts^*} \\
&= -\beta_{ws\cdot t}\sigma_{xx\cdot t}\check{\sigma}_{ss\cdot t}(\beta_{zs\cdot t} - \beta_{xs\cdot t}\beta_{zx\cdot t}) \\
&= -\beta_{ws\cdot t}\check{\sigma}_{ss\cdot t}\sigma_{xx\cdot t}\sigma_{zs\cdot xt}/\sigma_{ss\cdot t}.
\end{aligned}$$

By noting these results, equation (11) is derived. *Q.E.D.*

3.5 IDENTIFAIBLITY CRITERION: LATENT VARIABLE AND SELECTION BIAS CASE

Finally, we consider the case where both latent variables $\boldsymbol{U}$ and selection bias according to a selection criterion $a\leq S\leq b$ exist. Let $\boldsymbol{X}(X, Y\in\boldsymbol{X})$ and $\boldsymbol{T}$ be sets of observed variables, the steps for judging whether or not the total effect is identifiable are as follows:

Step 1: Check whether or not the combination of a subset of $\boldsymbol{X}\backslash\{X, Y\}$ and $\boldsymbol{U}\cup\boldsymbol{T}$ satisfies the back door criterion relative to (X, Y). If the answer is affirmative, go to Step 2.

Step 2: By noting that

$$\Sigma_{xx\cdot ts^*} = \Sigma_{xx\cdot t} - B_{xs\cdot t}B'_{xs\cdot t}\check{\sigma}_{ss\cdot t},$$

check whether or not the structure of zeros in $\Sigma_{xx\cdot t}$ (or $\Sigma^{-1}_{xx\cdot t}$) is such that every connectivity component of the complementary graph of $G^{x\cdot t}_{cov}$ (or $G^{x\cdot t}_{con}$) contains an odd cycle. If the answer is affirmative, since $\Sigma_{xx\cdot t}$ is identifiable (Stanghellini and Wermuth, 2005), then go to Step 3.

Step 3: Check whether Theorem 2 holds true for $\Sigma_{xx\cdot t}$ with regard to $\boldsymbol{U}$. If the answer is affirmative, since $\Sigma_{xx\cdot tu}$ is identifiable, then we can evaluate the total effect τ_{yx} of X on Y.

4 APPLICATION

The above results are applicable to analyze the data from a study about setting up painting conditions of car bodies, reported by Okuno et al. (1986). The data was collected with the purpose of setting up the process conditions, in order to increase transfer efficiency. The size of the sample is 38 and the variables of interest, each of which has zero mean and variance one, are the following:

Painting Condition : Dilution Ratio (X_1),
Degree of Viscosity (X_2), Painting Temperature (X_8)

Spraying Condition : Gun Speed (X_3),
Spray Distance (X_4), Atomizing Air Pressure (X_5),
Pattern Width (X_6), Fluid Output (X_7)

Environment Condition : Temperature (X_9),
Degree of Moisture (X_{10})

Response: Transfer Efficiency (Y)

Concerning this process, Kuroki et al. (2003) presented the path diagram shown in Fig. 5 (for the detail, see Kuroki et al., 2003). Based on the path diagram, Kuroki et al. (2003) presented the estimated correlation matrix. We here provide a part of the correlation matrix in Table 1. From Table 1, we assume that the covariance information on X_4 and X_{10} is not obtained.

Although $X_1, \cdots, X_6$ are considered to be controllable variables in Okuno et al. (1986), X_2 and X_6 are taken as treatment variables from controllable variables in order to evaluate their total effects from nonexperimental data in this paper.

Table 2 shows the selected variables for estimating total effects. The treatment variables of interest are listed in the first column. The second column shows

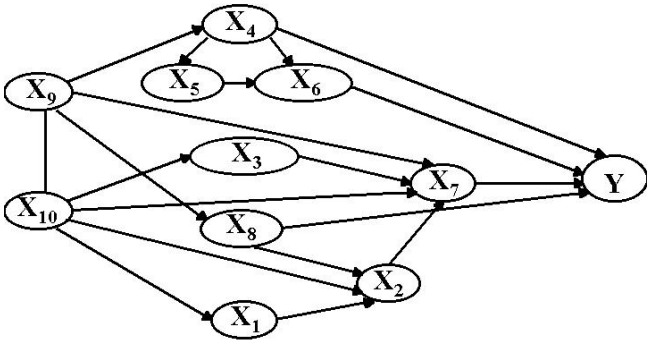


Fig. 5 : Path Diagram (Kuroki et al., 2003)

Table 1 : Estimated Correlation Matrix (Kuroki et al., 2003)

	X_1	X_2	X_5	X_6	X_8	X_9	Y
X_1	1.000	−0.736	0.028	−0.042	0.216	0.283	−0.091
X_2	−0.736	1.000	−0.063	0.095	−0.684	−0.635	0.326
X_5	0.028	−0.063	1.000	0.291	0.076	0.099	−0.277
X_6	−0.042	0.095	0.291	1.000	−0.114	−0.149	−0.250
X_8	0.216	−0.684	0.076	−0.114	1.000	0.761	−0.493
X_9	0.283	−0.635	0.099	−0.149	0.761	1.000	−0.475
Y	−0.091	0.326	−0.277	−0.250	−0.493	−0.475	1.000

Table 2 : Estimates of Total Effects

treatment	covariates	total effect
X_2	$Z = X_1$,$W = X_9$,$T = X_8$	-0.116
X_6	$Z = X_5$,$W = X_9$,$T = \phi$	-0.465

sets of covariates used for identifying total effects. The third columns shows the estimates of total effects. First, consider a situation that we wish to evaluate the total effect of X_2 on Y. Then, it can be recognized that the total effect can not be evaluated based on the back door criterion or the conditional IV method, because the covariance information on X_{10} can not be obtained from Table 2. In addition, since X_{10} exists in the back door path between X_2 and X_7, the front door criterion can not be applied, either. However, since a set of variables provided in the second column satisfies the conditions in Theorem 1, the total effect can be evaluated by using equation (10).

Next, consider a situation that we wish to evaluate the total effect of X_6 on Y. Then, it can be recognized that the total effect can not be evaluated based on the back door criterion or the conditional IV method, because the covariance information on X_4 can not be obtained from Table 2. In addition, there is not a set of variables satisfying the front door criterion. However, since a set of variables provided in the second column satisfies the conditions in Theorem 1, the total effect can be evaluated by using equation (10).

5 DISCUSSION

This paper discussed identification problems for total effects based on causal modeling in observational studies with latent variables and selection bias. In order to derive the graphical identifiability criteria, we introduced identification condition for factor models to the identification problem of total effects. In addition, we pointed out that there are some cases where the total effect is identifiable even when the identification condition for factor models does not hold true. Furthermore, we proposed new identification conditions of total effects, and provided the closed form expression of the identifiable total effects. The results of this paper help us judge from graph structure whether the total effect can be evaluated from observational studies in the presence of latent variables and selection bias.

ACKNOWLEGDEMENT

This research was supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan, the Kurata Foundation and the Mazda Foundation.

REFERENCES

Bowden, R. J. , and Turkington, D. A. (1984). *Instrumental Variables*, Cambridge University Press.

Brito, C. and Pearl, J. (2002). Generalized instrumental variables. *Proceeding of the 18th Conference on Uncertainty in Artificial Intelligence*, 85-93.

Cochran, W.G. (1938). The omission or addition of an independent variate in multiple linear regression, *Sup-*

plement to the Journal of the Royal Statistical Society, **5**, 171-176.

Cooper, G. F. (2000). A Bayesian method for causal modeling and discovery under selection. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, **16**, 98-106.

Greenland, S. (2003). Quantifying biases in causal models: Classical confounding versus collider-stratification bias. *Epidemiology*, **14**, 300-306.

Grzebyk, M., Wild, P. and Chouaniere, D. (2004). On identification of multi-factor models with correlated residuals. *Biometrika*, **91**, 141-151.

Hernan, M. A. , Hernandez-Diaz, S. and Robins, J. M. (2004). A structural approach to selection bias. *Epidemiology*, **15**, 615-625.

Huang, Y. and Valtorta, M. (2006). Pearl's Calculus of Intervention is Complete. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, **22**, 437-444.

Johnson, N. L. & Kotz, S. (1972). *Distributions in Statistics : Continuous Multivariate Distributions*. New York: John Wiley & Sons.

Kuroki, M.(2007). Identifiability Criteria for Total Effects in the Presence of Unmeasured Confounders (In Japanese),*Japanese Journal of Applied Statistics*,**36**, 71-85.

Kuroki, M. and Cai, Z. (2006). On Recovering Population's Covariance Matrix in the Presence of Selection Bias, *Biometrika*, **93**, 601-611.

Kuroki, M., Miyakawa, M. and Cai, Z. (2003). Joint Causal Effect in Linear Structural Equation Model and Its Application to Process Analysis, *Proceedings of the Workshop on Artificial Intelligence and Statistics*, **9**, 70-77.

Lauritzen, S. L. (1996). *Graphical models*, Clrendon Press, Oxford.

Okuno, T., Katayama, Z., Kamigori, N., Itoh, T., Irikura, N. and Fujiwara, N. (1986). *Kougyou ni okeru Tahenryou Data no Kaiseki* (In Japanese), Nikkagiren, Tokyo.

Pearl, J. (2000). *Causality: Models, reasoning, and inference*. Cambridge University Press.

Rao, C. R. (1973). *Linear statistical inference and its applications*. John Wiley & Sons.

Richardson, T. S. and Spirtes, P. (2002). Ancestral graph markov models. *Annals of Statistics*, **30**, 962-1030.

Shpitser,I. and Pearl, J. (2006). Identification of Joint Interventional Distributions in Recursive Semi-Markovian Causal Models. *Proceedings of the National Conference on Artificial Intelligence*,**21**, 1219-1226.

Spirtes, P., Glymour, C. and Scheines, R. (2000). *Causation, prediction, and search, 2nd edition*, MIT Press.

Spirtes, P. , Meek, C. , and Richardson, T. (1999). An algorithm for causal inference in the presence of latent variables and selection bias. In Glymour, C. and Cooper, G. , editors, *Computation, Causation, and Discovery*, 211-252.

Stanghellini, E. (1997). Identification of a single-factor models using graphical Gaussian rules. *Biometrika*, **84**, 241-244.

Stanghellini, E. (2004). Instrumental variables in Gaussian directed acyclic graph models with an unobserved confounder. *Environmetrics*, **15**, 463-469.

Stanghellini, E. and Wermuth, N. (2005). On the identification of directed acyclic graph models with one hidden variable. *Biometrika*, **92**, 337-350.

Tian, J. (2004). Identifying linear causal effects. *Proceeding of the Nineteenth National Conference on Artificial Intelligence*, 104-111.

Wermuth, N. (1989). Moderating effects in multivariate normal distributions. *Methodika*, **3**, 74-93.

Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics*, John Wiley and Sons.

Complexity of Inference in Graphical Models

Venkat Chandrasekaran
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

Nathan Srebro, Prahladh Harsha
Toyota Technological Institute – Chicago
Chicago, IL 60637

Abstract

It is well-known that inference in graphical models is hard in the worst case, but tractable for models with bounded treewidth. We ask whether treewidth is the only structural criterion of the underlying graph that enables tractable inference. In other words, is there some class of structures with unbounded treewidth in which inference is tractable? Subject to a combinatorial hypothesis due to Robertson et al. (1994), we show that low treewidth is indeed the *only* structural restriction that can ensure tractability. Thus, even for the "best case" graph structure, there is no inference algorithm with complexity polynomial in the treewidth.

1 INTRODUCTION

Graphical models offer a convenient representation for joint probability distributions over a large number of variables. Such models are defined as stochastic processes with respect to a graph: each vertex of the graph is associated with a random variable, and the edge structure specifies the conditional independence (Markov) properties among the variables. Due to their powerful modeling capabilities, graphical models have found applications in a variety of fields including computer vision (Szeliski, 1990), coding theory (McEliece et al., 1998), and statistical physics (Parisi, 1988). In many of these applications a commonly encountered problem is that of estimation or *inference*, which refers to computing the posterior marginal distribution of a variable at some vertex. We study the computational complexity of the inference problem in a graphical model consisting of discrete-valued variables as a function of structural properties of the underlying graph such as treewidth and planarity.

It is well-known that inference is **NP**-hard if no assumptions are made about the structure of the underlying graphical model (Cooper, 1990), and remains **NP**-hard even to approximate (Roth, 1996) — assuming $\mathbf{P} \neq \mathbf{NP}$, for any algorithm there are some structures in which (approximate) inference takes time super-polynomial in the size of the structure. However, inference in specific structures can still be tractable. For models in which the underlying graph has low treewidth, the junction-tree method provides an effective inference procedure that has complexity polynomial in the size of the graph, though exponential in the treewidth.

The notion of treewidth (Robertson and Seymour 1983; 1986) has led to several results in graph theory (Robertson et al., 1994) and to practical algorithms for a large class of **NP**-hard problems (Freuder, 1990). Among these problems is inference in graphical models, which, as mentioned earlier, can be solved in polynomial-time if the treewidth of the underlying graphs is bounded. For a triangulated graph the treewidth is one less than the size of the largest clique, while the treewidth of a general graph is the smallest treewidth over all triangulations of the graph.

A variety of results focus on special conditions on the potential functions (Fisher, 1966) that result in inference being tractable. Recently, some authors have attempted to provide *structural* conditions, aside from low treewidth, that allow efficient inference. For example, Chertkov and Chernyak (2006) describe an inference method based on enumeration over "generalized loops", suggesting that inference is tractable in graphs with a relatively small number of such loops.

In this paper we consider whether there might be an alternate structural property of graphs, which does not imply low treewidth, but guarantees tractable inference. Put differently we investigate the "best case", rather than worst case, hardness of inference with respect to the treewidth: does inference remain hard even in the "easiest" high-treewidth graph structures?

We focus only on structural properties, and consider algorithms that work (and are tractable) for any choice of the potentials.

Recently, Marx (2007) showed that constraint satisfaction problems (CSP) defined on any class of graphs with unbounded treewidth cannot be solved in polynomial-time. However, Marx's result refers only to algorithms for CSPs involving variables of *unbounded cardinality* (i.e. an unbounded number of states). Allowing such high-cardinality variables plays a critical role in the proof, which employs constructions using models in which the cardinality of the variables grows in an unbounded manner with the size of the graphs. Thus, translating these results to problems of inference in graphical models is of limited interest for typical inference problems. Marx's result can only imply hardness of inference when the number of states for each variable is unbounded, while most inference problems of interest involve variables with low cardinality or even binary states. Indeed, we usually think of the complexity of inference, or even of representation of a discrete graphical model, as growing exponentially with the number of states.

We focus on the complexity of inference in models consisting of *binary* variables defined on any class of graphs with unbounded treewidth. In such models a hardness result can be obtained if we assume a well-known hypothesis from graph minor theory. A minor of a graph $\mathcal{G}$ is a graph $\mathcal{H}$ that can be obtained from $\mathcal{G}$ by a sequence of vertex/edge deletions and/or edge contractions (see Section 2.4 for a precise definition). In a series of over twenty papers, Robertson and Seymour shed light on various aspects of graph minors and proved important results in graph theory. The theorem of greatest relevance to this paper is one that relates graph minors and treewidth: for each $g \times g$ grid-structured graph $\mathcal{G}$, there exists a finite $\kappa_{\mathrm{GM}}(g)$ such that $\mathcal{G}$ is a minor of *all* graphs with treewidth greater than $\kappa_{\mathrm{GM}}(g)$. The best known lower-bound and upper-bound for $\kappa_{\mathrm{GM}}(g)$ are $\Omega(g^2 \log g)$ and $2^{O(g^5)}$ respectively. The *grid-minor hypothesis* holds that $\kappa_{\mathrm{GM}}(g)$ is polynomial bounded with respect to g. The hypothesis is based on the belief that $\kappa_{\mathrm{GM}}(g)$ is closer to $\Omega(g^2 \log g)$ than $2^{O(g^5)}$ (Robertson et al., 1994); further evidence in support of this hypothesis is provided in Demaine et al. (2006).

Main result There is no class of graphical models consisting of *binary* variables with unbounded treewidth in which inference can be performed in time polynomial in the treewidth. The assumptions behind this result are that $\mathbf{NP} \not\subseteq \mathbf{P/poly}$[1] and the grid-minor hypothesis. Consequently, for *every* sequence of graphs $\{\mathcal{G}_k\}_{k=1}^{\infty}$ indexed by treewidth, inference is super-polynomial with respect to the treewidth k. More precisely, for *every* sequence of graphs $\{\mathcal{G}_k\}_{k=1}^{\infty}$ indexed by treewidth, there exists a choice of potential functions such that inference is super-polynomial with respect to treewidth k.

We also show the above result for planar graphs without requiring recourse to the grid-minor hypothesis — assuming *only* that $\mathbf{NP} \not\subseteq \mathbf{P/poly}$, there is no class of graphical models with binary variables defined on planar graphs with unbounded treewidth in which inference has complexity polynomial in the treewidth. We obtain sharper versions of these results that are based on the so-called exponential-time hypothesis (Impagliazzo et al., 2001) rather than the assumption that $\mathbf{NP} \not\subseteq \mathbf{P/poly}$. We further extend the hardness result above to hardness of approximation of the partition function to within an additive constant by a randomized polynomial-time algorithm.

Proof overview The standard "worst-case" hardness for the inference problem shows that there is *some* family of graphs $\{\mathcal{H}_k\}_{k=1}^{\infty}$ for which the inference problem is hard. In fact, it is known that the family of graphs can be assumed to be planar. To prove "best-case" hardness, we need to show that inference is hard in *every* family of graphs $\{\mathcal{G}_k\}_{k=1}^{\infty}$ of increasing treewidth. We prove this by showing that given *any* family of graphs $\{\mathcal{G}_k\}$ of increasing treewidth, we can reduce the general inference problem on planar graphs to the inference problem on $\{\mathcal{G}_k\}$. Our main tools are the Robertson-Seymour graph-minor theorems, which show that any planar graph is a minor of a not too large grid and a grid is a minor of any graph with not too large treewidth. We use these results to reduce the inference problem on any planar graph to one on a grid of not too large size and then to any graph (in particular $\mathcal{G}_k$) of not too large treewidth (k). Unfortunately, the known theoretical guarantees on the "not too large treewidth" is too weak (in fact, super-exponential) for our purposes. We get around this problem by either relying on the grid-minor hypothesis which conjectures that the "not too large treewidth" is in fact at most polynomial in the size of the grid, or by assuming that $\{\mathcal{G}_k\}$ is a family of planar graphs. This completes our proof but for one caveat: it is not sufficient if we know that any planar graph is a minor of the grid and the grid is a minor of $\mathcal{G}_k$; we actually need the sequence of minor operations between these graphs. In general, finding the sequence of minor operations that transform one arbitrary graph to another is an **NP**-complete problem. However, it is known

[1] The assumption $\mathbf{NP} \not\subseteq \mathbf{P/poly}$ is the non-uniform version of the more popular $\mathbf{NP} \neq \mathbf{P}$ assumption. For more details on uniform vs. non-uniform algorithms, see Section 2.2.

that a planar graph can be embedded in a grid in linear time (Tamassia & Tollis, 1989). This solves one of our two problems. For the other (embedding a grid into $\mathcal{G}_k$), we exploit the fact that both these graphs are fixed and depend only on the size of the input instance and not on the actual instance itself. Hence, we could non-uniformly hardwire this sequence of minor operations into our algorithm and thus obtain a non-uniform hardness reduction.

Organization The rest of this paper is organized as follows. Section 2 provides a brief background on inference in graphical models, treewidth, and graph minors. Section 3 presents the formal statement of the problem addressed in this paper. Section 4 describes constraint satisfaction problems; we prove a reduction from such problems to inference in graphical models that plays a key role in our analysis. Section 5 provides the main results of this paper. We conclude with a brief discussion and open questions in Section 6.

2 BACKGROUND

2.1 GRAPHICAL MODELS AND INFERENCE

A *graph* $\mathcal{G} = (V, \mathcal{E})$ consists of a set of vertices V and associated edges $\mathcal{E} \subset \binom{V}{2}$, where $\binom{V}{2}$ is the set of all unordered pairs of vertices. A *graphical model* (Lauritzen, 1996) is a collection of random variables indexed by the vertices of a graph; each vertex $v \in V$ corresponds to a random variable x_v, and where for any $A \subset V$, $x_A = \{x_v | v \in A\}$. We assume that each of the variables x_v is discrete-valued with cardinality q. Of interest in this paper are distributions that factor according to a graph $\mathcal{G} = (V, \mathcal{E})$ as follows:

$$p(x_V) = \frac{1}{Z(\psi)} \prod_{v \in V} \psi_v(x_v) \prod_{E \in \mathcal{E}} \psi_E(x_E). \quad (1)$$

Here, each ψ_E (or ψ_v) is only a function of the variables x_E (or variable x_v). The functions ψ_v and ψ_E are non-negative and are also known as *potential* or *compatibility* functions. We denote the collection of these potentials by $\psi = \{\psi_v, v \in V\} \cup \{\psi_E, E \in \mathcal{E}\}$. The function $Z(\psi)$ is called the *partition* function and serves to normalize the distribution:

$$Z(\psi) = \sum_{x_V \in \{0, \cdots, q-1\}^{|V|}} \prod_{v \in V} \psi_v(x_v) \prod_{E \in \mathcal{E}} \psi_E(x_E). \quad (2)$$

Given a posterior distribution that factors according to a graph as described above, a common task in applications such as image processing and computer vision (Szeliski, 1990) is to compute the marginal distribution of a variable at some vertex. It is well-known that the complexity of computing the marginal distribution at some vertex is comparable to that of computing the partition function. A polynomial-time procedure to solve one of these problems can be used to construct a polynomial-time algorithm for the other. Thus, we consider in this paper the complexity of computing the partition $Z(\psi)$, and with an abuse of terminology, it is this problem that we refer to as *inference*. The intractability of inference arises due to the fact that there are exponentially many terms in the sum in (2). We study the complexity of inference as a function of structural properties of the underlying graph.

2.2 UNIFORM VS. NON-UNIFORM ALGORITHMS

The classical notion of algorithms refers to "uniform algorithms" in which one has a single algorithm that works for all input lengths. A "non-uniform algorithm" on the other hand refers to a family of algorithms, one for each input length. Another way of thinking about such non-uniform algorithms is that the algorithm is allowed to receive some arbitrary oracle advice that depends only on the input length (but not on the actual input). In the theory of computation literature, such "non-uniform algorithms" are usually referred to as fixed-input-size "circuits", where for each input length a different circuit is used. The class **P** is the class of problems that have polynomial time uniform algorithms while **P/poly** is its non-uniform counterpart, i.e., the class of problems that have polynomial time non-uniform algorithms (circuits). Clearly, $\mathbf{P} \subset \mathbf{P/poly}$. The non-uniform version of the assumption $\mathbf{NP} \neq \mathbf{P}$ is $\mathbf{NP} \not\subseteq \mathbf{P/poly}$, and (though slightly weaker) is equally believed to be true. We need to work with the latter assumption since our proof proceeds by reducing the "best-case" inference problem to a non-uniform algorithm for **NP**.

2.3 GRAPH TREEWIDTH

A graph is said to be *triangulated* if every cycle of length greater than three contains an edge between two non-adjacent vertices. The *treewidth* $\mathrm{tw}(\mathcal{G})$ of a triangulated graph $\mathcal{G}$ is one less than the size of the largest clique. The treewidth of a general graph is defined

$$\mathrm{tw}(\mathcal{G}) = \min_{\mathcal{H} \supseteq \mathcal{G}, \mathcal{H} \text{ triangulated}} \mathrm{tw}(\mathcal{H}).$$

Here, $\mathcal{H} \supseteq \mathcal{G}$ denotes that $\mathcal{H}$ is a supergraph of $\mathcal{G}$. In words, the treewidth of a graph $\mathcal{G}$ is the minimum over the treewidths of all triangulated supergraphs of $\mathcal{G}$.

Figure 1 shows an example of a non-triangulated graph $\mathcal{G}$, which has a 4-cycle with no edge connecting non-

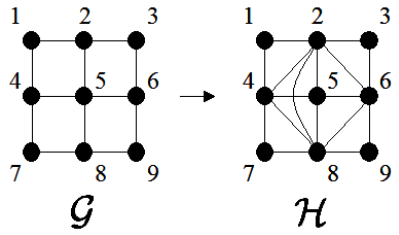


Figure 1: A non-triangulated graph $\mathcal{G}$, and a triangulated supergraph $\mathcal{H}$ of $\mathcal{G}$.

adjacent vertices. The graph $\mathcal{H}$ is a triangulated supergraph of $\mathcal{G}$, and has treewidth 3 as the largest clique is $\{2, 5, 6, 8\}$. Thus, the treewidth of $\mathcal{G}$ is also 3.

The complexity of a graphical model is often measured by the treewidth of the underlying graph. Distributions defined on trees, which are treewidth-1, permit very efficient linear-time inference algorithms. For loopy graphs that have low treewidth the junction-tree method (Cowell et al., 1999) provides an efficient inference algorithm. However, for general loopy graphs the junction-tree method might be intractable because it scales exponentially with the treewidth. As a result considerable effort is being devoted to the development of approximate inference algorithms. Our focus here is on the computational complexity of exact (or near-exact) inference.

2.4 GRAPH MINORS

The theory of graph minors plays a key role in our analysis. Specifically, we show in Section 2.4.1 that the complexity of inference in a minor of $\mathcal{G}$ is bounded by the complexity of inference in $\mathcal{G}$. A *minor* of a graph is obtained by any sequence of the following operations:

- **Vertex deletion**: Given a graph $(V, \mathcal{E})$, a vertex $v \in V$ is *deleted*, as are all the edges $\mathcal{E}_v = \{E \in \mathcal{E} : v \in E\}$ incident on v, to obtain the graph $(V \backslash v, \mathcal{E} \backslash \mathcal{E}_v)$.
- **Edge deletion**: Given a graph $(V, \mathcal{E})$, an edge $E \in \mathcal{E}$ is *deleted* to obtain the graph $(V, \mathcal{E} \backslash E)$.
- **Edge contraction**: Given a graph $(V, \mathcal{E})$, an edge $\{u, v\} \in \mathcal{E}$ is *contracted* to form a single vertex u' with edges to every vertex in $V \backslash \{u, v\}$ that previously had an edge to either u or v. Thus, the resulting graph has one less vertex than the original graph.

Figure 2 gives an example of each of these operations. The graph $\mathcal{H}_1$ is a minor of $\mathcal{G}$, and is obtained from $\mathcal{G}$ by deleting the edge $\{5, 6\}$. Next, $\mathcal{H}_2$ is obtained from

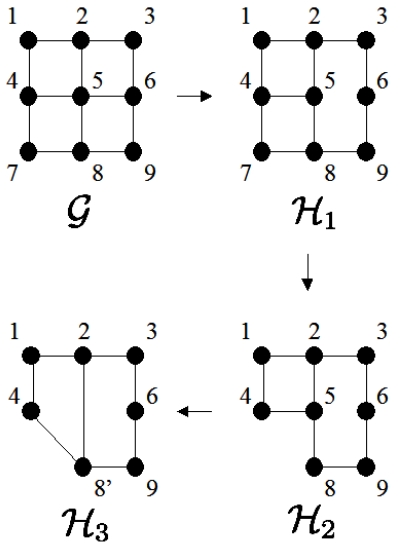


Figure 2: A graph $\mathcal{G}$, and three of its minors $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$ obtained by edge deletion, followed by vertex deletion, and finally edge contraction.

$\mathcal{H}_1$ by deleting the vertex 7, and the corresponding edges $\{4, 7\}, \{7, 8\}$ that are incident on 7. Thus, $\mathcal{H}_2$ is a minor of both $\mathcal{H}_1$ and $\mathcal{G}$. Finally, $\mathcal{H}_3$ is obtained from $\mathcal{H}_2$ by contracting the edge $\{5, 8\}$ to form the new vertex $8'$, which now has an edge to vertices 2, 4, and 9. The graph $\mathcal{H}_3$ is a minor of each of the graphs $\mathcal{G}$, $\mathcal{H}_1$, and $\mathcal{H}_2$.

In a series of over twenty papers, Robertson and Seymour investigated various aspects of graph minors and proved several important results in graph theory. The following theorem played a key role in proving many of these results; it provides a connection between treewidth and graph minors, and forms a critical component of our analysis.

Theorem 2.1. *(Robertson et al., 1994) Let $\mathcal{G}$ be a $g \times g$ grid. There exists a* finite $\kappa_{\mathrm{GM}}(g)$ *such that $\mathcal{G}$ is a minor of* all *graphs with treewidth greater than $\kappa_{\mathrm{GM}}(g)$. Further, the best known bounds on $\kappa_{\mathrm{GM}}(g)$ are that $c_1 g^2 \log g \leq \kappa_{\mathrm{GM}}(g) \leq 2^{c_2 g^5}$, where c_1 and c_2 are universal constants (i.e. they are independent of g).*

Thus, each grid-structured graph is a minor of all graphs with sufficiently large treewidth. Robertson et al. (1994) expressed the belief that $\kappa_{\mathrm{GM}}(g)$ is closer to $c_1 g^2 \log g$ than $2^{c_2 g^5}$, and may even be on the order of $g^2 \log g$. In addition, Demaine et al. (2006) build further support for this belief and conjecture that $\kappa_{\mathrm{GM}}(g) \sim g^3$. Consequently, we have the following grid-minor hypothesis.

Grid-minor hypothesis: $\kappa_{\mathrm{GM}}(g)$, as defined in Theorem 2.1, is polynomial in g.

This hypothesis is a key assumption in the proof of our 'Main Result' as stated in the introduction. Next, we state a restricted result that relates graph minors and treewidth for planar graphs. A planar graph (Bollobás, 1998) is one that can be drawn on a plane with no two edges intersecting each other.

Theorem 2.2. *(Robertson et al., 1994) There exist universal constants c_3 and c_4 such that the following holds. Let $\mathcal{G}$ be a $g \times g$ grid. Then, (a) $\mathcal{G}$ is a minor of all planar graphs with treewidth greater than $c_3 g$. Further, (b) all planar graphs of size (number of vertices) less than $c_4 g$ are minors of $\mathcal{G}$.*

Hence, Theorem 2.2 states that $\kappa_{\mathrm{GM}}(g)$ is actually *linear* in g for planar graphs.

2.4.1 Inference and graph minors

Let $\mathcal{M}(\mathcal{G}, q)$ refer to the set of all possible choices for potential functions on the vertices and edges of $\mathcal{G} = (V, \mathcal{E})$, with the variables having maximum cardinality q. That is, each $\psi \in \mathcal{M}(\mathcal{G}, q)$ is specified as $\psi = \{\psi_v, v \in V\} \cup \{\psi_E, E \in \mathcal{E}\}$. In the following lemma, we relate the complexity of inference in a minor of a graph $\mathcal{G}$ to inference in $\mathcal{G}$.

Lemma 2.3. *Let $\mathcal{H}$ be a minor of $\mathcal{G}$, and let $\psi_{\mathcal{H}} \in \mathcal{M}(\mathcal{H}, q)$. There exists a $\psi_{\mathcal{G}} \in \mathcal{M}(\mathcal{G}, q)$ such that $Z(\psi_{\mathcal{H}}) = Z(\psi_{\mathcal{G}})$. Moreover, $\psi_{\mathcal{G}}$ can be computed in linear time given $\psi_{\mathcal{H}}$ and the sequence of minor operations that transform $\mathcal{G}$ to $\mathcal{H}$.*

Proof. All we need to show is that if a graph $\mathcal{H} = (V_{\mathcal{H}}, \mathcal{E}_{\mathcal{H}})$ is obtained from another graph $\mathcal{G} = (V_{\mathcal{G}}, \mathcal{E}_{\mathcal{G}})$ by just a *single* application of one of the standard minor operations, then we can transform a given $\psi_{\mathcal{H}} \in \mathcal{M}(\mathcal{H}, q)$ into a $\psi_{\mathcal{G}} \in \mathcal{M}(\mathcal{G}, q)$ with $Z(\psi_{\mathcal{G}}) = Z(\psi_{\mathcal{H}})$.

Vertex deletion: Suppose that $v \in V_{\mathcal{G}}$ as well as edges $\mathcal{E}_v \subseteq \mathcal{E}_{\mathcal{G}}$ that are incident on v in $\mathcal{G}$ are deleted. Let $\psi_v = \frac{1}{q}$ and let $\psi_E = 1, \forall E \in \mathcal{E}_v$. Letting $\psi_{\mathcal{G}} = \cup_{E \in \mathcal{E}_v} \psi_E \cup \psi_v \cup \psi_{\mathcal{H}}$, one can check that $Z(\psi_{\mathcal{G}}) = Z(\psi_{\mathcal{H}})$.

Edge deletion: Suppose that $E \in \mathcal{E}_{\mathcal{G}}$ is deleted. Setting $\psi_E = 1$, and $\psi_{\mathcal{G}} = \psi_{\mathcal{H}} \cup \psi_E$, one can check that $Z(\psi_{\mathcal{G}}) = Z(\psi_{\mathcal{H}})$.

Edge contraction: Suppose that $\{u, v\} \in \mathcal{E}_{\mathcal{G}}$ is contracted to form the new vertex $u' \in V_{\mathcal{H}}$. We define $\psi_{\{u,v\}}(x_u, x_v) = \delta(x_u - x_v)$, where $\delta(\cdot)$ is the Kronecker delta function that evaluates to 1 if the argument is 0, and 0 otherwise. For the edge potentials, if a vertex $w \in V_{\mathcal{G}} \backslash \{u, v\}$ is originally connected in $\mathcal{G}$ by an edge to only one of u or v, then we set the corresponding $\psi_{\{u,w\}}$ or $\psi_{\{v,w\}}$ to be equal to $\psi_{\{u',w\}}$. If *both* u and v are originally connected by edges to w in $\mathcal{G}$, then we define $\psi_{\{u,w\}} = \psi_{\{u',w\}}$ and $\psi_{\{v,w\}} = 1$. Finally, we define the vertex potentials as $\psi_u = \psi_{u'}$ and $\psi_v = 1$. Letting all the other vertex and edge potentials in $\mathcal{G}$ be the same as those in $\mathcal{H}$, it is easily seen that $Z(\psi_{\mathcal{G}}) = Z(\psi_{\mathcal{H}})$. □

Thus, an inference problem in a minor of $\mathcal{G}$ can be transformed to an inference problem in $\mathcal{G}$. Consequently, this result allows us to establish hardness of inference in a graph $\mathcal{G}$, by establishing hardness of inference in a minor of $\mathcal{G}$.

3 PROBLEM STATEMENT

Let $T_f(I)$ denote the runtime of an algorithm f on input I. We consider inference algorithms that take as input a graph $\mathcal{G} = (V, \mathcal{E})$ and an element of $\mathcal{M}(\mathcal{G}, q)$ (i.e., potentials defined with respect to the vertices and edges of $\mathcal{G}$), and compute the partition function $Z(\psi)$. We would like to investigate the impact of the treewidth $\mathrm{tw}(\mathcal{G})$ of the graph $\mathcal{G}$ on the required runtime of any inference algorithm.

Typical complexity analysis studies the worst case, or maximum, runtime of an algorithm over all inputs. Since inference in a graphical model is **NP**-hard, and assuming $\mathbf{NP} \neq \mathbf{P}$, we know that the worst case runtime of any inference algorithm must scale super-polynomially with the size of the graph. That is, the *maximum* runtime over all graphs is super-polynomial.

Our focus in this paper is on studying the following "best case" complexity of inference:

$$\beta_f(k, q) = \min_{\mathcal{G}:\mathrm{tw}(\mathcal{G})=k} \max_{\psi \in \mathcal{M}(\mathcal{G},q)} T_f(\mathcal{G}, \psi). \qquad (3)$$

In words, $\beta_f(k, q)$ captures the complexity of inference as a function of treewidth by finding the "best", or "easiest" graph of treewidth k for each k. Since we are primarily concerned with bounds that are independent of the cardinality q, we will specifically consider the case $q = 2$ and define $\mathcal{M}(\mathcal{G}) = \mathcal{M}(\mathcal{G}, 2), \beta_f(k) = \beta_f(k, 2)$.

Main Question: How does $\beta_f(k)$, as defined in (3), grow as a function of the treewidth k for any inference algorithm f? Does there exist an inference algorithm f for which $\beta_f(k)$ grows only polynomially with k?

If there exists an f such that $\beta_f(k)$ is polynomial in k, then there exists a class of structures with unbounded treewidth in which inference would be tractable. Alternatively, if $\beta_f(k)$ is not polynomial in k for any procedure f, then bounding the treewidth is the only structural restriction on graphical models that leads to tractable inference.

The quantity $\beta_f(k)$ in the 'Main Question' refers to a uniform algorithm, i.e. a single algorithm that should work for graphs of all treewidths. However, to answer this question we will actually study a slightly harder question, where we allow non-uniform algorithms specialized to a sequence of graphs of increasing treewidths. Given a sequence of graphs $\{\mathcal{G}_k\}_{k=1}^{\infty}$ with $\text{tw}(\mathcal{G}_k) = k$, we will analyze the runtime of any (non-uniform) sequence $f = \{f_k\}_{k=1}^{\infty}$ of algorithms (i.e. a "non-uniform algorithm"), where f_k solves the inference problem on $\mathcal{G}_k$. For any such sequence, we study how the runtime increases (taking worst case over potential functions) with k, i.e. $\max_{\psi \in \mathcal{M}(\mathcal{G}_k)} T_{f_k}(\mathcal{G}_k, \psi)$ as a function of k. Taking the infimum over the choice of the sequence of graphs $\{\mathcal{G}_k\}_{k=1}^{\infty}$ (i.e. choosing the "easiest" sequence of graphs of increasing treewidth) gives a lower bound on $\beta_f(k)$.

Our 'Main Question' pertains to *exact* inference. We also investigate the tractability of obtaining an approximation to the partition function. Specifically, we consider the problem of computing $Z(\psi)$ up to an additive constant ε, perhaps using a randomized procedure. Focusing on (randomized) algorithms $f(\mathcal{G}, \psi, \varepsilon)$ that provide a $\widehat{Z}$ such that $Z(\psi) - \varepsilon \leq \widehat{Z} \leq Z(\psi) + \varepsilon$ (with high probability—see Section 5.2), we consider:

$$\beta_f^{\varepsilon}(k) = \min_{\mathcal{G}:\text{tw}(\mathcal{G})=k} \max_{\psi \in \mathcal{M}(\mathcal{G})} T_f(\mathcal{G}, \psi, \varepsilon). \qquad (4)$$

Note that $\mathcal{M}(\mathcal{G}) = \mathcal{M}(\mathcal{G}, 2)$. We can now ask a question analogous to our 'Main Question', for $\beta_f^{\varepsilon}(k)$ rather than $\beta_f(k)$.

4 CONSTRAINT SATISFACTION AND INFERENCE

A *constraint satisfaction problem* (CSP) is defined as a set of constraints specified on subsets of a collection of discrete-valued variables. Each constraint is said to be *satisfied* for some stipulated configurations of the variables in the constraint. The problem is to identify a configuration of the variables that satisfies all the constraints (i.e., find a *satisfying* assignment). We will consider CSP as a decision problem — the problem of deciding if such a satisfying assignment exists. We will mostly be concerned with 2-CSPs: CSPs in which each constraint involves only two variables. Note that one can associate a graph with an instance of a 2-CSP, with the vertices representing the variables and edges present only between those vertices that appear in the same constraint. A related problem is the MAX CSP in which one is interested in configurations of the variables that *maximize* the number of satisfied constraints. Again, we will refer to MAX CSP as the problem of deciding, for some integer d, if there are any configurations that simultaneously satisfy more that d constraints.

An important special case of a CSP is the SAT problem, in which disjunctive constraints are specified on binary variables. Although polynomial time algorithms exist for 2-SAT, the MAX 2-SAT problem is **NP**-complete. In fact we have that planar MAX 2-SAT, in which instances are restricted to those defined on planar graphs, is also **NP**-complete (Guibas et al., 1991).

To obtain sharper results, we will use the so-called "Exponential-Time Hypothesis" (Impagliazzo et al., 2001):

Exponential-time hypothesis (ETH): There exists no non-uniform algorithm[2] that can solve arbitrary instances of n-variable 3-SAT in time $2^{o(n)}$.

Note that $\mathbf{NP} \not\subseteq \mathbf{P/poly}$ would merely state that there exists no polynomial-time (non-uniform) algorithm for arbitrary n-variable instances of 3-SAT (since 3-SAT is **NP**-complete). Thus, the ETH is a stronger assumption than $\mathbf{NP} \not\subseteq \mathbf{P/poly}$, and consequently, allows one to obtain sharper bounds on the growth of $\beta_f(k)$ and $\beta_f^{\varepsilon}(k)$ (see Section 5 for more details).

In order to translate hardness results for CSPs and MAX CSPs to the problem of inference in graphical models, we prove the following lemma that transforms instances of 2-CSPs to inference problems in graphical models. Specifically, we show that each instance of a MAX 2-CSP can be mapped to a particular decision-version of an inference problem.

Lemma 4.1. *Let $I = (x_1, \cdots, x_n; \mathcal{R})$ be an instance of a MAX 2-CSP problem, where $x_1, \cdots, x_n$ are discrete-valued variables (of cardinality q) and $\mathcal{R}$ is a set of constraints. Let $\mathcal{G} = (V, \mathcal{E})$ denote the graph which represents the instance I. There exist a set of potentials $\psi \in \mathcal{M}(\mathcal{G}, q)$ and a function $h : \{0, \cdots, |\mathcal{R}|\} \to \mathbb{R}^+$ with the following property[3]: at least d disjunctions in $\mathcal{R}$ can be satisfied simultaneously if and only if $Z(\psi) \geq h(d)$. Moreover, the construction of the potentials ψ and the evaluation of the function h are polynomial-time operations, given I.*

Proof. Let $\mathcal{G} = (V, \mathcal{E})$ denote the graph which represents the instance I. Hence, $|V| = n$ with each variable being assigned to a vertex and $\mathcal{E}$ contains only those pairs of vertices for which the corresponding variables

[2] Impagliazzo et al. (2001) refer to the uniform version of ETH, but their results equally apply to the above-stated non-uniform version of the hypothesis, which is also widely believed to be true.

[3] $|\mathcal{R}|$ denotes the number of constraints in $\mathcal{R}$.

appear in the same relation, so that $|\mathcal{E}| = |\mathcal{R}|$. For each $E \in \mathcal{R}$, define

$$\psi_E(x_E) = \begin{cases} 1, & x_E \text{ satisfies } E \\ \varepsilon, & \text{otherwise.} \end{cases}$$

Define vertex potentials similarly for each vertex constraint, and set $\psi_v = 1$ for other vertices. Choose $\varepsilon \in (0, \frac{1}{q^n})$. Let $h(d) = \varepsilon^{|\mathcal{E}|-d}$. If I is such that at least d constraints can be simultaneously satisfied, it is clear that $Z(\psi) \geq \varepsilon^{|\mathcal{E}|-d}$. Alternatively, if I is such that d or more constraints can never be satisfied simultaneously, then we have that

$$Z(\psi) \leq \sum_{x_V \in \{0,\cdots,q-1\}^n} \varepsilon^{|\mathcal{E}|-d+1} = q^n \varepsilon h(d) < h(d).$$

□

By setting $d = |\mathcal{R}|$ in the above lemma, one can transform instances of a 2-CSP problem to a decision-version of an inference problem. Next, we translate the recent result in (Marx, 2007) for 2-CSPs to a complexity result for inference.

Theorem 4.2. *(Marx, 2007)*[4] *Let $\{\mathcal{G}_k\}_{k=1}^{\infty}$ be* any *sequence of graphs indexed by treewidth. Suppose that there exists an algorithm g for instances of 2-CSPs, with variables of arbitrary cardinality, defined on the graphs $\mathcal{G}_k$. Let $q(\psi)$ be the maximum cardinality of a variable referred to by the constraints ψ. If $T_g(\mathcal{G}_k, \psi) = q(\psi)^{o(\frac{k}{\log k})}$, then the ETH fails.*

Corollary 4.3. *Let f be any algorithm that can perform inference on graphical models with variables of arbitrary cardinality. Under the ETH, for any $r(k) = o(k/\log k)$ there exist q, k such that $\beta_f(k, q) > q^{r(k)}$.*

A consequence of this corollary is that the junction-tree algorithm (Cowell et al., 1999), which scales as q^k, is in a sense near-optimal (assuming the ETH). However, as we noted in the introduction, this result has a significant weakness in that it provides an asymptotic lower bound only for sufficiently large cardinalities. It does not provide a lower bound for any fixed cardinality q. This restriction plays an important role in the reductions in (Marx, 2007), in which large sets of variables in an intermediate model are represented using a single high-cardinality variable. In the following section, we describe our main results for the complexity of inference in graphical models with *binary* variables, which are typically of most interest to the machine learning community.

Another important class of problems arising from constraint satisfaction is that of *counting* the number of satisfying assignments in a CSP. Such counting problems are titled #CSPs, and planar #2-SAT falls under the class of #**P**-complete problems (Vadhan, 2001). Instances of these problems can also be transformed to inference in graphical models.

Lemma 4.4. *Let $I = (x_1, \ldots, x_n; \mathcal{R})$ be an instance of a #2-CSP problem, where $x_1, \ldots, x_n$ are discrete-valued variables of cardinality q and $\mathcal{R}$ is a set of constraints. Let $\mathcal{G}$ be the graph which represents the instance I. There exists a set of potentials $\psi \in \mathcal{M}(\mathcal{G}, q)$ such that the number of satisfying assignments is $\lfloor Z(\psi) \rfloor$. Moreover, the construction of the potentials ψ is a polynomial-time operation, given I.*

Proof. The construction of the potentials is similar to that in the proof of Lemma 4.1. One can check that the resulting partition function has the property that the integer part is equal to the number of satisfying solutions. □

In the following section, we use the #**P**-completeness of planar #2-SAT to demonstrate the hardness of approximation of $Z(\psi)$ up to an additive constant.

5 MAIN RESULTS

We present our main results for graphical models with binary-valued variables in this section.

5.1 EXACT INFERENCE

Theorem 5.1. *Let $\{\mathcal{G}_k\}_{k=1}^{\infty}$ be an infinite sequence of graphs indexed by treewidth. Let $f = \{f_k\}_{k=1}^{\infty}$ be any (possibly non-uniform) sequence of algorithms that solves the inference problem on $\{\mathcal{G}_k\}$ with binary variables. Furthermore, let $T(k)$ denote the worst-case running time of f on $\mathcal{G}_k$ (i.e, $T(k) = \max_{\psi \in \mathcal{M}(\mathcal{G}_k)} T_{f_k}(\mathcal{G}_k, \psi)$).*

(a) Assuming that **NP** $\not\subseteq$ **P/poly** *and that the grid-minor hypothesis holds, $T(k)$ is super-polynomial in k. Hence, $\beta_f(k)$ as defined in (3) is super-polynomial with respect to k.*

(b) Assuming that $\kappa_{\mathrm{GM}}(g) = O(g^r)$ in the grid-minor hypothesis and the ETH, we have that $T(k) = 2^{\Omega(k^{1/2r})}$. Hence, $\beta_f(k) = 2^{\Omega(k^{1/2r})}$.

Proof. (a) Suppose (for contradiction) that there exists a (possibly non-uniform) polynomial time algorithm f that solves the inference problem on $\{\mathcal{G}_k\}_{k=1}^{\infty}$. More precisely, let $f = \{f_k\}_{k=1}^{\infty}$ be a sequence of algorithms such that f_k solves the inference problem on $\mathcal{G}_k$ in polynomial time. Assuming the grid-minor hypothesis, we will demonstrate that this implies a

[4] The statement here is actually of a non-uniform variant of the result of Marx (2007).

non-uniform polynomial time algorithm for the inference problem on any planar graph. Recall that planar MAX 2-SAT is **NP**-complete (Guibas et al., 1991) and polynomial-time reducible to the inference problem on planar graphs (Lemma 4.1). This provides a (non-uniform) polynomial time algorithm for an **NP**-complete problem, contradicting the $\mathbf{NP} \not\subseteq \mathbf{P/poly}$ assumption.

Given an instance $(\mathcal{G}, \psi)$ of the inference problem on planar graphs, we proceed as follows: Let $|\mathcal{G}| = s$. By Theorem 2.2, $\mathcal{G}$ is a minor of the $s/c_4 \times s/c_4$ grid. Furthermore, the sequence of minor operations that transform a $s/c_4 \times s/c_4$ grid to $\mathcal{G}$ can be obtained in polynomial time (Tamassia & Tollis, 1989). Thus, using Lemma 2.3, the inference problem $(\mathcal{G}, \psi)$ can be reduced to an inference problem on the $s/c_4 \times s/c_4$ grid in time linear in s. By Theorem 2.1, the $s/c_4 \times s/c_4$ grid is a minor of $\mathcal{G}_{\kappa_{\rm GM}(s/c_4)}$. We will now use as "non-uniform advice" the sequence of minor operations that transform $\mathcal{G}_{\kappa_{\rm GM}(s/c_4)}$ to the $s/c_4 \times s/c_4$ grid. Note that this depends only on the input size s and not on the actual instance $(\mathcal{G}, \psi)$. Using Lemma 2.3 again, we can reduce the inference problem on the $s/c_4 \times s/c_4$ grid to an inference problem on $\mathcal{G}_{\kappa_{\rm GM}(s/c_4)}$ in linear time. We now use $f_{\kappa_{\rm GM}(s/c_4)}$ to solve the inference problem on $\mathcal{G}_{\kappa_{\rm GM}(s/c_4)}$, thus solving the original inference problem $(\mathcal{G}, \psi)$. The fact that $T(k)$, and thus also the size of the graph $\mathcal{G}_k$, is at most polynomial in k and the grid-minor hypothesis (i.e, $\kappa_{\rm GM}(g) = \text{poly}(g)$) imply that the above algorithm is a polynomial time (non-uniform) algorithm for the inference problem on planar graphs.

(b) We obtain the tighter hardness result by carefully analyzing the running time of the inference algorithm on planar graphs suggested in (a). It can be easily checked that the above algorithm runs in time $T(\kappa_{\rm GM}(s/c_4))$, which is $T(O(s^r))$ if $\kappa_{\rm GM}(g) = O(g^r)$. Combining this with the reduction from 3-SAT to planar MAX 2-SAT ((Litchenstein, 1982; Guibas et al., 1991)), which blows up the instance size by a quadratic factor, we obtain a $T(O(n^{2r}))$ time non-uniform algorithm for n-variable instances of 3-SAT. Recall that the ETH states there exists no (non-uniform) algorithm for arbitrary n-variable instances of 3-SAT that has running time $2^{o(n)}$. Hence, assuming the grid-minor hypothesis and the ETH, we must have that $T(O(n^{2r}))$ is at least $2^{O(n)}$ or equivalently that $T(n)$ is at least $2^{\Omega(n^{1/2r})}$. □

Theorem 5.1 provides an answer to the 'Main Question' in Section 3, and comprises the 'Main Result' described in the introduction. Notice that the ETH assumption enables a sharper performance bound instead of the simpler result that $\beta_f(k)$ is super-polynomial in k (of part (a)). Next, we have the following theorem for planar graphs that does *not* require assumption of the grid-minor hypothesis. Define

$$\beta_f^{\mathsf{planar}}(k) = \min_{\mathcal{G}:\text{tw}(\mathcal{G})=k,\ \mathcal{G}\ \text{planar}} \ \max_{\psi \in \mathcal{M}(\mathcal{G})} T_f(\mathcal{G}, \psi). \quad (5)$$

Theorem 5.2. *Let f be any inference algorithm that operates on graphical models with binary variables defined on planar graphs.*

(a) Assuming that $\mathbf{NP} \not\subseteq \mathbf{P/poly}$, $\beta_f^{\mathsf{planar}}(k)$ as defined in (5) is super-polynomial with respect to k.

(b) Under the ETH, $\beta_f^{\mathsf{planar}}(k) = 2^{\Omega(k^{1/2})}$.

Proof. The proof is similar to that of Theorem 5.1, but the grid-minor hypothesis is not required due to Theorem 2.2. □

Based on the results in (Demaine et al., 2005) Theorem 5.2 holds more generally for the inference problem in graphical models defined on bounded-genus graphs, of which planar graphs are a special case.

5.2 APPROXIMATE INFERENCE

The above results show that exact inference is intractable in any class of graphs with unbounded treewidth. Here, we prove that even obtaining an approximation within some additive constant to the partition function is intractable. Our result uses Lemma 4.4 along with the fact that planar #2-SAT is #**P**-complete (Vadhan, 2001).

Theorem 5.3. *Suppose that the grid-minor hypothesis holds. Let f be any (randomized) approximate inference algorithm that operates on graphical models with binary variables. Let $\varepsilon > 0$ and $0 < \delta < 1$ be specified, and suppose that $f(\mathcal{G}, \psi, \varepsilon)$ provides an approximation $\widehat{Z}$ such that*

$$\Pr[Z(\psi) - \varepsilon \leq \widehat{Z} \leq Z(\psi) + \varepsilon] \geq 1 - \delta.$$

If $\beta_f^{\varepsilon}(k)$, as defined in (4), is polynomial in k, $\frac{1}{\varepsilon}$, and $\log(\frac{1}{\delta})$, then $\mathbf{NP} \subseteq \mathbf{P/poly}$.

Proof. Unless $\mathbf{NP} \subseteq \mathbf{P/poly}$, there is no randomized non-uniform procedure that can approximate the solution to a #**P**-complete problem to within a constant c with probability greater than $1 - \delta$, which is polynomial in the size of the problem, c, and $\log(\frac{1}{\delta})$ (Vazirani, 2004). Based on Lemma 4.4, we have that instances of planar #2-SAT can be reduced to performing inference in a model defined on a planar graph so that the number of solutions is equal to $\lfloor Z(\psi) \rfloor$. Using the fact that planar #2-SAT is #**P**-complete, one can prove this result by following the same line of analysis adopted in the proof of Theorem 5.1. □

6 CONCLUSION

With increasing interest in understanding various inference procedures and providing conditions under which they are correct and tractable, it is important to understand whether there might indeed be some structural property, other than treewidth, which can guarantee tractable inference. In this paper we studied this issue, presenting and discussing the relevant literature from the CSP community as well as relevant graph theory concepts, and can conclude that it is not likely such an alternate property exists—finding a property that ensures tractability of inference without bounding treewidth would imply providing a counterexample to Robertson and Seymour's grid-minor hypothesis.

We believe that relating the "best case" complexity of inference to the grid-minor hypothesis provides substantial evidence that inference remains hard even in the "easiest" high-treewidth graph structures. Nevertheless, it would be of great interest to prove the results in this paper without resorting to the grid minor hypothesis. It would also be useful to obtain hardness results that are valid even for reasonably restricted classes of potential functions, e.g. potential functions with bounded dynamic range.

Acknowledgements We would like to thank Lance Fortnow and Jaikumar Radhakrishnan for helpful discussions and referring us to (Tamassia & Tollis, 1989).

References

Bollobás, B. (1998). *Modern graph theory*, vol. 184 of *Graduate Texts in Mathematics*. Springer Verlag.

Chertkov, M., & Chernyak, V. Y. (2006). Loop series for discrete statistical models on graphs. *Journal of Statistical Mechanics*, *2006*, P06009. doi:10.1088/1742-5468/2006/06/P06009.

Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, *42*, 393–405. doi:10.1016/0004-3702(90)90060-D.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., & Spiegelhalter, D. J. (1999). *Probabilistic networks and expert systems*. Springer.

Demaine, E. D., Fomin, F. V., Hajiaghayi, M. T., & Thilikos, D. M. (2005). Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *Journal of the ACM*, *52*, 866–893. doi:10.1145/1101821.1101823.

Demaine, E. D., Hajiaghayi, M. T., & Kawarabayashi, K. (2006). Algorithmic graph minor theory: Improved grid minor bounds and Wagner's contraction. *Proc.* 17*th International Symposium on Algorithms and Computation (ISAAC)* (pp. 3–15). Kolkata, India: Springer. doi:10.1007/11940128_3.

Fisher, M. E. (1966). On the dimer solution of planar Ising models. *Journal of Mathematical Physics*, *7*, 1776–1781. doi:10.1063/1.1704825.

Freuder, E. C. (1990). Complexity of k-tree structured constraint satisfaction problems. *Proc.* 8*th National Conference on Artificial Intelligence (AAAI)* (pp. 4–9). Boston, Massachusetts. `http://www.aaai.org/Library/AAAI/1990/aaai90-001.php`.

Guibas, L. J., Hershberger, J., Mitchell, J. S. B., & Snoeyink, J. (1991). Approximating polygons and subdivisions with minimum link paths. *Proc.* 2*nd International Symposium on Algorithms (ISA)* (pp. 151–162). Taipei, Republic of China: Springer. doi:10.1007/3-540-54945-5_59.

Impagliazzo, R., Paturi, R., & Zane, F. (2001). Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, *63*, 512–530. doi:10.1006/jcss.2001.1774.

Lauritzen, S. L. (1996). *Graphical models*. Oxford University Press.

Litchenstein, D. (1982). Planar formulae and their uses. *SIAM Journal of Computing*, *11*, 329–343. doi:10.1137/0211025.

Marx, D. (2007). Can you beat treewidth? *Proceedings of the* 48*th IEEE Symposium on Foundations of Computer Science (FOCS)* (pp. 169–179). Providence, Rhode Island. doi:10.1109/FOCS.2007.4389490.

McEliece, R. J., MacKay, D. J. C., & Cheng, J.-F. (1998). Turbo decoding as an instance of Pearl's "Belief Propagation" Algorithm. *IEEE Journal on Selected Areas in Communications*, *16*, 140–152. doi:10.1109/49.661103.

Parisi, G. (1988). *Statistical field theory*. Addison-Wesley.

Robertson, N., & Seymour, P. D. (1983). Graph minors. I. Excluding a forest. *Journal of Combinatorial Theory, Series B*, *35*, 39–61. doi:10.1016/0095-8956(83)90079-5.

Robertson, N., & Seymour, P. D. (1986). Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, *7*, 309–322. doi:10.1016/0196-6774(86)90023-4.

Robertson, N., Seymour, P. D., & Thomas, R. (1994). Quickly excluding a planar graph. *Journal of Combinatorial Theory, Series B*, *62*, 323–348. doi:10.1006/jctb.1994.1073.

Roth, D. (1996). On the hardness of approximate reasoning. *Artificial Intelligence*, *82*, 273–302. doi:10.1016/0004-3702(94)00092-1.

Szeliski, R. (1990). Bayesian modeling of uncertainty in low-level vision. *International Journal of Computer Vision*, *5*, 271–301. doi:10.1007/BF00126502.

Tamassia, R., & Tollis, I. G. (1989). Planar grid embedding in linear time. *IEEE Transactions on Circuits and Systems*, *36*, 1230–1234. doi:10.1109/31.34669.

Vadhan, S. P. (2001). The complexity of counting in sparse, regular, and planar graphs. *SIAM Journal on Computing*, *31*, 398–427. doi:10.1137/S0097539797321602.

Vazirani, V. V. (2004). *Approximation algorithms*. Springer.

Approximating the Partition Function by Deleting and then Correcting for Model Edges

Arthur Choi and **Adnan Darwiche**
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
{*aychoi,darwiche*}*@cs.ucla.edu*

Abstract

We propose an approach for approximating the partition function which is based on two steps: (1) computing the partition function of a simplified model which is obtained by deleting model edges, and (2) rectifying the result by applying an edge-by-edge correction. The approach leads to an intuitive framework in which one can trade-off the quality of an approximation with the complexity of computing it. It also includes the Bethe free energy approximation as a degenerate case. We develop the approach theoretically in this paper and provide a number of empirical results that reveal its practical utility.

1 INTRODUCTION

We presented in prior work an approach to approximate inference which is based on performing exact inference on a simplified model (Choi & Darwiche, 2006a, 2006b). We proposed obtaining the simplified model by deleting enough edges to render its treewidth manageable under the current computational resources. Interestingly enough, the approach subsumes iterative belief propagation (IBP) as a degenerate case, and provides an intuitive framework for capturing a class of Generalized Belief Propagation (GBP) approximations (Choi & Darwiche, 2006a; Yedidia, Freeman, & Weiss, 2005).

We show in this paper that the simplified models can also be used to approximate the partition function if one applies a correction for each deleted edge. We propose two edge-correction schemes, each of which is capable of perfectly correcting the partition function when a single edge has been deleted. The first scheme will have this property only when a particular condition holds in the simplified model, and gives rise to the Bethe free energy approximation when applied to a tree-structured approximation (see Yedidia et al., 2005, for more on the Bethe approximation and its relationship to IBP). The second correction scheme does not require such a condition and is shown empirically to lead to more accurate approximations. Both schemes can be applied to the whole spectrum of simplified models and can therefore be used to trade-off the quality of obtained approximations with the complexity of computing them.

This new edge-correction perspective on approximating the partition function has a number of consequences. First, it provides a new perspective on the Bethe free energy approximation, and may serve as a tool to help identify situations when Bethe approximations may be exact or accurate in practice. Next, it suggests that we do not necessarily need to seek good approximations, but instead seek approximations that are accurately correctable. To this end, we propose a heuristic for finding simplified models that is specific to the task of correction. Finally, it provides the opportunity to improve on edge-deletion approximations (and certain GBP approximations), with only a modest amount of computational effort. In particular, we show empirically how it is possible to correct only for a small number of edges that have the most impact on an approximation.

Proofs of results appear in the Appendix.

2 EDGE DELETION

We first review our edge deletion framework in probabilistic graphical models. For simplicity, we consider pairwise Markov random fields, although our framework can easily be extended to general Markov networks as well as to factor graphs. For an application to directed models, see (Choi & Darwiche, 2006a).

Let a pairwise Markov random field (MRF) $\mathcal{M}$ have a graph $(\mathcal{E}, \mathcal{V})$ with edges $(i, j) \in \mathcal{E}$ and nodes $i \in \mathcal{V}$,

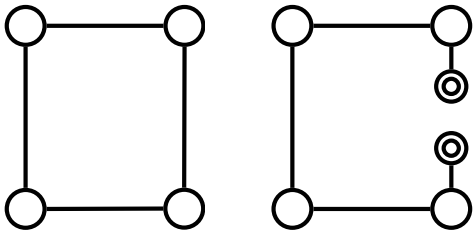


Figure 1: An MRF (left); after edge deletion (right).

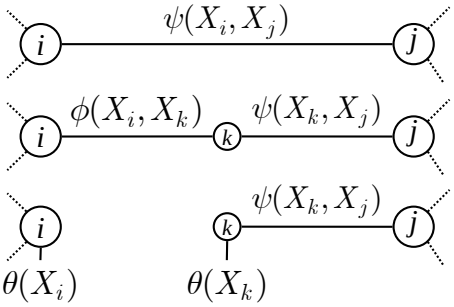


Figure 2: To delete edge (i, j) (top), we introduce auxiliary node k (middle), and delete equivalence edge (i, k), adding edge parameters (bottom).

where each node i of the graph is associated with a variable X_i taking on values x_i. Edges (i, j) are associated with edge potentials $\psi(x_i, x_j)$ and nodes i with node potentials $\psi(x_i)$. The (strictly positive) distribution Pr induced by $\mathcal{M}$ is defined as follows:

$$Pr(\mathbf{x}) \stackrel{def}{=} \frac{1}{Z} \prod_{(i,j)\in\mathcal{E}} \psi(x_i, x_j) \prod_{i\in\mathcal{V}} \psi(x_i),$$

where $\mathbf{x}$ is an instantiation $x_1, \ldots, x_n$ of network variables, and where Z is the *partition function:*

$$Z \stackrel{def}{=} \sum_{\mathbf{x}} \prod_{(i,j)\in\mathcal{E}} \psi(x_i, x_j) \prod_{i\in\mathcal{V}} \psi(x_i).$$

The basic idea behind our framework is to delete enough edges from the pairwise MRF to render it tractable for exact inference.

Definition 1 *Let $\mathcal{M}$ be a pairwise MRF. To delete edge (i, j) from $\mathcal{M}$ we remove the edge (i, j) from $\mathcal{M}$ and then introduce the auxiliary potentials $\theta(X_i)$ and $\theta(X_j)$ for variables X_i and X_j.*

Figure 1 provides an example of deleting an edge. When deleting multiple edges, note that we may introduce multiple, yet distinct, potentials $\theta(X_i)$ for the same node X_i. We shall refer to auxiliary potentials $\theta(X_i)$ and $\theta(X_j)$ as *edge parameters* and use Θ to denote the set of all edges parameters. The resulting pairwise MRF will be denoted by $\mathcal{M}'(\Theta)$, its partition function will be denoted by $Z'(\Theta)$ and its distribution will be denoted by $Pr'(.; \Theta)$. When choosing a particular value for edge parameters Θ, we will drop reference to Θ, using only $\mathcal{M}'$, Z' and $Pr'(.)$.

Note that while the distribution $Pr(.)$ and partition function Z of the original pairwise MRF $\mathcal{M}$ may be hard to compute, the distribution $Pr'(.; \Theta)$ and partition function $Z'(\Theta)$ of $\mathcal{M}'(\Theta)$ should be easily computable due to edge deletion. Note also that before we can use $Pr'(.; \Theta)$ and $Z'(\Theta)$ to approximate $Pr(.)$ and Z, we must first specify the edge parameters Θ. In fact, it is the values of these parameters which will control the quality of approximations $Pr'(.; \Theta)$ and $Z'(\Theta)$.

Without loss of generality, we will assume that we are only deleting *equivalence edges* (i, j), which connect two variables X_i and X_j with the same domain, and have a potential $\phi(x_i, x_j)$ that denotes an equivalence constraint: $\phi(x_i, x_j) = 1$ if $x_i = x_j$, and $\phi(x_i, x_j) = 0$ otherwise. The deletion of any edge in an MRF can be formulated as the deletion of an equivalence edge.[1]

As for the values of the edge parameters, we proposed (and justified) in (Choi & Darwiche, 2006a) the following conditions on $\theta(x_i)$ and $\theta(x_j)$:

$$\theta(x_i) = \alpha \frac{\partial Z'}{\partial \theta(x_j)} \quad \text{and} \quad \theta(x_j) = \alpha \frac{\partial Z'}{\partial \theta(x_i)} \qquad (1)$$

where α is a normalizing constant. Note that the partial derivatives of Equation 1 can be computed efficiently in traditional inference frameworks (Darwiche, 2003; Park & Darwiche, 2004).

Equation 1 can also be viewed as update equations, suggesting an iterative method that searches for edge parameters, which we called ED-BP (Choi & Darwiche, 2006a). Starting with an initial approximation $\mathcal{M}'_0$ at iteration $t = 0$ (say, with uniform parameters), we can compute edge parameters $\theta_t(x_i)$ and $\theta_t(x_j)$ for an iteration $t > 0$ by performing exact inference in the approximate network $\mathcal{M}'_{t-1}$. We repeat this process until we observe that all parameters converge to a fixed point satisfying Equation 1 (if ever).

Note that Equation 1 does not specify a unique value of edge parameters, due to the constants α. That is, each value of these constants will lead to a different set of edge parameters. Yet, independent of which constants we use, the resulting pairwise MRF $\mathcal{M}'$ will

[1] To delete an MRF edge (i, j) that is not an equivalence edge, we use the technique illustrated in Figure 2: we introduce an auxiliary node k between i and j; introduce an equivalence constraint on the edge (i, k); copy the original potential of edge (i, j) to (k, j); and delete the equivalence edge (i, k). Note that the original model and the extended one will: (1) have the same treewidth, (2) agree on the distribution over their common variables, and (3) have the same partition function values.

have an *invariant* distribution $Pr'(.)$ that satisfies the following properties. First,

$$Pr'(x_i) = Pr'(x_j) = \frac{1}{z_{ij}} \cdot \theta(x_i)\theta(x_j), \qquad (2)$$

where $z_{ij} = \sum_{x_i=x_j} \theta(x_i)\theta(x_j)$. Next, if the pairwise MRF $\mathcal{M}'$ has a tree structure, the node and edge marginals of distribution $Pr'(.)$ will correspond precisely to the marginals obtained by running IBP on the original model $\mathcal{M}$. Moreover, if the pairwise MRF $\mathcal{M}'$ has loops, the node marginals of distribution Pr' will correspond to node marginals obtained by running generalized belief propagation (GBP) using a particular joingraph for the original model $\mathcal{M}$ (Yedidia et al., 2005; Choi & Darwiche, 2006a).

3 EDGE CORRECTION

While the edge parameters specified by Equation 1 are guaranteed to yield an invariant distribution $Pr'(.)$, they are not guaranteed to yield an invariant partition function Z' as this function is sensitive to the choice of constants α. Hence, while these edge parameters will yield an interesting approximation of node marginals, they do not yield a meaningful approximation of the partition function.

We will show in this section, however, that one can apply an edge-by-edge correction to the partition function Z', leading to a corrected partition function that is invariant to the choice of constants α. This seemingly subtle approach leads to two important consequences. First, it results in a semantics for the Bethe free energy approximation as a corrected partition function. Second, it allows for an improved class of approximations based on improved corrections.

3.1 ZERO EDGE-CORRECTION

We will now propose a correction to the partition function Z', which gives rise to the Bethe free energy and some of its generalizations.

Proposition 1 *Let $\mathcal{M}'$ be the result of deleting a single equivalence edge (i, j) from a pairwise MRF $\mathcal{M}$. If the parameters of edge (i, j) satisfy Equation 1, and if the mutual information between X_i and X_j in $\mathcal{M}'$ is zero, then:*

$$Z = Z' \cdot \frac{1}{z_{ij}}, \quad \textit{where} \quad z_{ij} = \sum_{x_i=x_j} \theta(x_i)\theta(x_j).$$

That is, if we delete a *single* edge (i, j) and find that X_i and X_j are independent in the resulting model $\mathcal{M}'$, we can correct the partition function Z' by z_{ij} and recover the exact partition function Z. Moreover, the result of this correction is invariant to the constants α used in Equation 1.

From now on, we will use $MI(X_i; X_j)$ to denote the mutual information between two variables X_i and X_j, computed in the *simplified* MRF $\mathcal{M}'$. Moreover, when $MI(X_i; X_j) = 0$, we will say that the deleted edge (i, j) is a zero-MI edge. Note that while an edge may be zero-MI in $\mathcal{M}'$, the mutual information between X_i and X_j in the original MRF $\mathcal{M}$ may still be high.

Let us now consider the more realistic situation where we delete multiple edges, say $\mathcal{E}^\star$, from $\mathcal{M}$ to yield the model $\mathcal{M}'$. We propose to accumulate the above correction for each of the deleted edges, leading to a corrected partition function $Z' \cdot \frac{1}{z}$, where

$$z = \prod_{(i,j)\in\mathcal{E}^\star} z_{ij} = \prod_{(i,j)\in\mathcal{E}^\star} \sum_{x_i=x_j} \theta(x_i)\theta(x_j). \qquad (3)$$

We will refer to this correction as a *zero-MI edge correction*, or EC-Z. This correction is no longer guaranteed to recover the exact partition function Z, even if each of the deleted edges is a zero-MI edge. Yet, if the pairwise MRF $\mathcal{M}'$ has a tree structure, applying this correction to the partition function Z' gives rise to the Bethe free energy approximation.

To review, the Bethe free energy F_β, is an approximation of the true free energy F of a pairwise MRF $\mathcal{M}$, and is exact when $\mathcal{M}$ has a tree structure (Yedidia et al., 2005). In this case, $F = -\log Z$, so we can in principle use F_β as an approximation of the partition function Z, even when $\mathcal{M}$ does not have a tree structure, i.e., we can use $Z_\beta = \exp\{-F_\beta\}$.

Theorem 1 *Let $\mathcal{M}'$ be the result of deleting equivalence edges from a pairwise MRF $\mathcal{M}$. If $\mathcal{M}'$ has a tree structure and its edge parameters are as given by Equation 1, we have $Z_\beta = Z' \cdot \frac{1}{z}$.*

Hence, the Bethe approximation of Z is a degenerate case of the EC-Z correction. Thus, IBP and the closely related Bethe approximation, which are exact when an MRF $\mathcal{M}$ is a tree, are naturally characterized by tree-structured ED-BP approximations $\mathcal{M}'$. In particular, exact inference in the simplified network $\mathcal{M}'$ yields: (1) node and edge marginals that are precisely the approximate marginals given by IBP (Choi & Darwiche, 2006a), and now (2) a rectified partition function that is precisely the Bethe approximation; cf. (Wainwright, Jaakkola, & Willsky, 2003).[2]

[2]Wainwright et al. proposed tree-based reparametrization (TRP), an algorithm that iteratively reparameterizes the node and edge potentials of a pairwise MRF. At convergence, the node and edge potentials of a tree (any tree)

Since the EC-Z correction is specified purely in quantities available in the model $\mathcal{M}'$, it will be easily computable as long as the model $\mathcal{M}'$ is sparse enough (i.e., it has a treewidth that is manageable under the given computational resources). Hence, this correction can be practically applicable even if $\mathcal{M}'$ does not have a tree structure. In such a case, the correction will lead to an approximation of the partition function which is superior to the one obtained by the Bethe free energy. We will illustrate this point empirically in Section 6.

3.2 GENERAL EDGE-CORRECTION

Proposition 1 gives us a condition that allows us to correct the partition function exactly, but under the assumption that the single edge deleted is zero-MI. The following result allows us, in fact, to correct the partition function when deleting *any* single edge.

Proposition 2 *Let $\mathcal{M}'$ be the result of deleting a single equivalence edge (i, j) from a pairwise MRF $\mathcal{M}$. If the parameters of edge (i, j) satisfy Equation 1, then:*

$$Z = Z' \cdot \frac{y_{ij}}{z_{ij}}, \quad \text{where} \quad y_{ij} = \sum_{x_i = x_j} Pr'(x_i \mid x_j).$$

Note that when the deleted edge (i, j) happens to be zero-MI, factor y_{ij} is 1, and thus Proposition 2 reduces to Proposition 1.

We can also use this proposition as a basis for correcting the partition function when multiple edges are deleted, just as we did in Equation 3. In particular, we now propose using the correction $Z' \cdot \frac{y}{z}$, where z is the same factor given in Equation 3, and

$$y = \prod_{(i,j) \in \mathcal{E}^\star} y_{ij} = \prod_{(i,j) \in \mathcal{E}^\star} \sum_{x_i = x_j} Pr'(x_i \mid x_j), \quad (4)$$

which we refer to as a *general edge correction*, or EC-G.

We note that when every edge is deleted in an ED-BP network $\mathcal{M}'$, every deleted edge becomes a zero-MI edge. Thus, in this case, EC-G reduces to EC-Z, and both yield the Bethe free energy, as in Theorem 1. As we recover more edges, we may expect EC-G to offer improved approximations over EC-Z, as it relaxes the zero-MI assumption for deleted edges. Accordingly, we may want to delete different edges for EC-G than we would for EC-Z.

embedded in the reparametrized MRF induces a distribution whose exact node and edge marginals are consistent with the corresponding marginals given by IBP. In contrast to ED-BP, TRP's embedded-tree distributions are already normalized, i.e., their partition function is 1. Moreover, generalizations of TRP appeal to auxiliary representations, via reparametrization in joingraphs and hypergraphs. In contrast, the semantics of ED-BP suggest that we simply delete fewer edges. As we shall see in Section 5, the semantics of edge correction further suggest intuitive edge recovery heuristics for choosing more structured approximations.

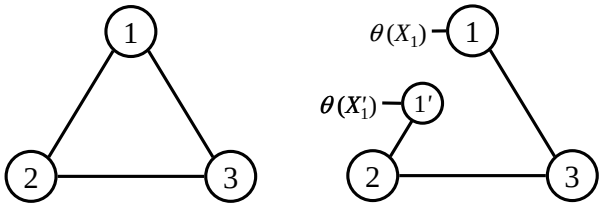


Figure 3: An MRF (left); after deleting edge $(1, 2)$, as in Figure 2 (right).

4 AN EXAMPLE

We provide here an illustrative example of our edge correction techniques. Consider a network of three nodes X_1, X_2 and X_3 that form a clique, with the following edge potentials:

X_i	X_j	$\psi(X_1, X_2)$	$\psi(X_1, X_3)$	$\psi(X_2, X_3)$
x_i	x_j	.9	.1	.081
x_i	$\bar{x}_j$	.1	.9	.810
$\bar{x}_i$	x_j	.1	.9	.090
$\bar{x}_i$	$\bar{x}_j$	.9	.1	.900

Suppose now that we delete the edge $(1, 2)$ by replacing $(1, 2)$ with a chain $\{(1, 1'), (1', 2)\}$ and deleting the equivalence edge $(1, 1')$; see Figure 3. Using ED-BP to parameterize this deleted edge, we have (to 4 digits):

X_i	$\theta(X_1)$	$\theta(X_1')$
x_i	.4789	.8273
$\bar{x}_i$	.5211	.1727

and we compute $Z' \approx 0.4447$. In this example, edge $(1, 1')$ happens to be a zero-MI edge, so $y_{ij} = 1$ and $z_{ij} \approx 0.4862$. Further, we know that both Propositions 1 and 2 allow us to recover the true partition function $Z = Z' \cdot \frac{1}{z_{ij}} \approx 0.9146$.

Now, suppose that we replace the potential on edge $(2, 3)$ with $1 - \psi(X_2, X_3)$. In this case, ED-BP gives us edge parameters (to 4 digits):

X_i	$\theta(X_1)$	$\theta(X_1')$
x_i	.5196	.1951
$\bar{x}_i$	.4804	.8049

and we compute $Z' \approx 0.5053$. In this case, edge $(1, 1')$ is not a zero-MI edge. Here, we find that $y_{ij} \approx 1.0484$ and $z_{ij} \approx 0.4880$. Since we only delete a single edge, Proposition 2 recovers the true partition function $Z = Z' \cdot \frac{y_{ij}}{z_{ij}} = 1.08542$ whereas Proposition 1 gives only an approximation $Z' \cdot \frac{1}{z_{ij}} \approx 1.0353$.

5 EDGE RECOVERY

Suppose we already have a tree-structured approximation $\mathcal{M}'$ of the original model $\mathcal{M}$, but are afforded more computational resources. We can then improve the approximation by *recovering* some of the deleted edges. However, which edge's recovery would have the most impact on the quality of the approximation?

Edge Recovery for EC-Z. Since EC-Z is exact for a single deleted edge when $MI(X_i; X_j) = 0$, one may want to recover those edges (i, j) with the highest mutual information $MI(X_i; X_j)$. In fact, this is the same heuristic proposed by (Choi & Darwiche, 2006a) for improving marginal approximations. We will indeed show the promise of this heuristic for EC-Z corrections, in Section 6. On the other hand, we also show that it turns out to be a poor heuristic for EC-G corrections.

Edge Recovery for EC-G. Consider the situation when two equivalence edges are deleted, (i, j) and (s, t). In this case, we use the approximate correction:

$$Z' \cdot \frac{y}{z} = Z' \cdot \frac{y_{ij}}{z_{ij}} \frac{y_{st}}{z_{st}},$$

where $\frac{y_{ij}}{z_{ij}}$ is the single-edge correction for edge (i, j) and $\frac{y_{st}}{z_{st}}$ is the single-edge correction for edge (s, t).

The question now is: When is this double-edge correction exact? Intuitively, we want to identify a situation where each edge can be corrected, independently of the other. Consider then the case where variables X_i, X_j are independent of variables X_s, X_t in $\mathcal{M}'$.

Proposition 3 *Let $\mathcal{M}'$ be the result of deleting two equivalence edges, (i, j) and (s, t), from a pairwise MRF $\mathcal{M}$. If the edge parameters of $\mathcal{M}'$ satisfy Equation 1, and if $MI(X_i X_j; X_s X_t) = 0$ in $\mathcal{M}'$, then:*

$$Z = Z' \cdot \frac{y_{ij}}{z_{ij}} \frac{y_{st}}{z_{st}}.$$

This suggests a new edge recovery heuristic for EC-G approximations to the partition function. Initially, we start with a tree-structured network $\mathcal{M}'$. We assign each deleted edge (i, j) a score:

$$\sum_{(s,t) \in \mathcal{E}^\star \setminus (i,j)} MI(X_i X_j; X_s X_t).$$

We then prefer to recover the top k edges with the highest mutual information scores.

6 EXPERIMENTS

Our goal here is to highlight different aspects of edge-correction, edge-recovery, and further a notion of partial correction. Starting from a random spanning tree

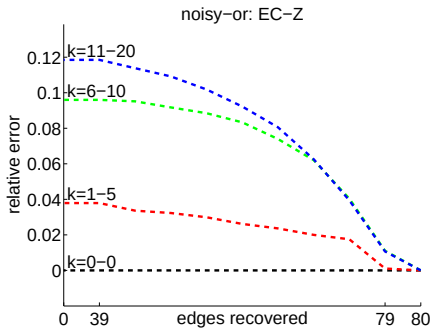


Figure 4: Edge correction in noisy-or networks.

(dropping instances where ED-BP and hence IBP, do not converge), we rank each deleted edge, and recover edges k at a time until all edges are recovered. At each point, we evaluate the quality of the approximation by the average relative error $|\widehat{Z} - Z|/Z$, where $\widehat{Z}$ denotes the designated approximation. Remember that in a tree-structured approximation, when no edge is recovered, EC-Z corresponds to the Bethe approximation. Likewise, when every edge is recovered, both EC-Z and EC-G are exact. Although, for simplicity, we presented our edge-correction framework in the context of pairwise MRFs, some of our experiments are run on Bayesian networks, to which all of our results also apply.[3] In these cases, observations are generated from the joint distribution over all leaves, unless otherwise specified.

Noisy-or. We consider first random two-layer noisy-or networks. Deleting an edge in this network effectively disconnects a cause variable C from an effect variable E, where a clone $\hat{C}$ replaces C as a a cause of E.[4] In this situation, we may use edge-correction to reason how well EC-Z and the Bethe approximation may perform. With no positive findings, for example, we know that all causes are pairwise mutually independent, including a cause C and its clone $\hat{C}$ in a noisy-or network where edges have been deleted. Starting from a tree-structured approximation, corresponding to the Bethe approximation, every recoverable edge is zero-MI and will remain zero-MI up to the point where all edges are recovered. Thus we may infer EC-Z to be exact throughout, and thus also that the Bethe approximation is exact.

Consider now Figure 4, which compares the quality of EC-Z corrections as edges are recovered randomly. We generated over 400 random noisy-or networks,[5] where

[3] Most of the Bayesian networks used here are available at http://www.cs.huji.ac.il/labs/compbio/Repository.

[4] As in Section 2, we replace edge $C \to E$ with a chain $C \to \hat{C} \to E$, and delete the equivalence edge $C \to \hat{C}$.

[5] Each network was given 20 roots and 20 sinks, where

for each network, we randomly chose k of 20 effect variables as positive findings and the remaining effect variables as negative findings. We have 4 cases here measuring the quality of the EC-Z approximation, each an average over a range of positive findings: 0, 1–5, 6–10, 11–20. As predicted, the EC-Z and Bethe approximations are exact with 0 positive findings. Given this, we expect, and observe, that with more positive findings, and fewer zero-MI edges, the EC-Z and Bethe approximations tend to be less accurate.

Edge recovery. Consider now Figure 5, where we compare EC-Z corrections to EC-G corrections, but also the impact that different edge recovery heuristics can have on an approximation. Here, plots are averages of over 50 instances. In the first plot, we took random 6×6 grid networks, where pairwise couplings were randomly given parameters in $[0.0, 0.1)$ or $(0.9, 1.0]$. First, when we compare EC-Z and EC-G by random edge recovery, we see that EC-G is a notable improvement over EC-Z, even when no edges are recovered. When we use the mutual information heuristic (MI) designed for EC-Z, the EC-Z approximations also improve considerably. However, EC-G approximations are worse than when we randomly recovered edges! Although EC-G approximations still dominate the EC-Z ones, this example illustrates that EC-Z approximations (based on the Bethe approximation) and EC-G approximations (based on exact corrections for a single edge) are of a different nature, and suggest that an alternative approach to recovery may be needed. Indeed, when we use the mutual information heuristic (MI2) designed for EC-G, we find that EC-G easily dominates the first four approximations. We see similar results in the `win95pts` and `water` networks.

Partial corrections. Although the individual edge-corrections for EC-Z are trivial to compute, the corrections for EC-G require joint marginals. In the case where we need to correct for many deleted edges, the EC-G corrections of Equation 4 may become expensive to compute. We may then ask: Can we effectively improve an approximation, by correcting for only a subset of the edges?

Consider then Figure 6, where we plot how the quality of our approximation evolves over time (averaged over 50 instances), over two steps: (1) the ED-BP parametrization algorithm, and after convergence (2) EC-G edge correction. On the first half of each plot, we start with a tree-structured approximate network, and compute the EC-Z approximation as ED-BP (and equivalently, IBP, in this case) runs for a fixed number of iterations. Eventually, the edge-corrected partition function converges (to the Bethe approximation), at which point we want to compute the edge corrections for EC-G. We can compute the corrections for an edge, one-by-one, applying them to the EC-Z approximation as they are computed. Since edge corrections are invariant to the order in which they are computed, we can then examine a notion of a *partial* EC-G approximation that accumulates only the correction factors for a given subset of deleted edges.

On the right half of each plot, we compute the error in a partial EC-G approximation given two separate orderings of deleted edges. The first ordering, which we consider to be "optimal", pre-computes corrections for all edges and sorts them from largest to smallest. In the `win95pts` network, we find that in fact, most of the edges have very little impact on the final EC-G approximation! Moreover, the time it took to compute the most important corrections required only as much time as it took ED-BP (IBP) to converge. This suggests that it is possible to improve on the Bethe approximation, with only a modest amount of additional computation (in the time to compute corrections for the important edges).

Of course, such an approach would require a way to identify the most important corrections, without actually computing them. In (Choi & Darwiche, 2008), we proposed a soft extension of d-separation in polytree Bayesian networks that was shown to be effective in ranking edges for the process of edge recovery (as in EC-Z). Applying it here to the task of ranking edge-corrections, we find that it is also effective at identifying important edges for correction. For example, in the `win95pts` network, soft d-separation (sd-sep) is nearly as competitive with the "optimal" at producing partial EC-G approximations. Moreover, soft d-separation is much more efficient, requiring only node and edge marginals to rank *all* deleted edges.

We see a similar story in the `pigs` and `mildew` network. In the `mildew` network, where many deleted edges have an impact on the approximation, the quality of the approximation tends to improve monotonically (on average), so we may still desire to perform as many individual corrections as resources allow.

7 EDGE CORRECTIONS AND FREE ENERGIES

As the Bethe free energy is an edge-corrected partition function, EC-Z and EC-G approximations can be viewed also from the perspective of free energies.

When the model $\mathcal{M}'$ is a tree, EC-Z yields the influential Bethe free energy approximation (Yedidia et al., 2005). When the model $\mathcal{M}'$ has cycles, it can be shown that EC-Z corresponds more generally to jo-

sinks are given 4 random parents. Network parameters were also chosen randomly.

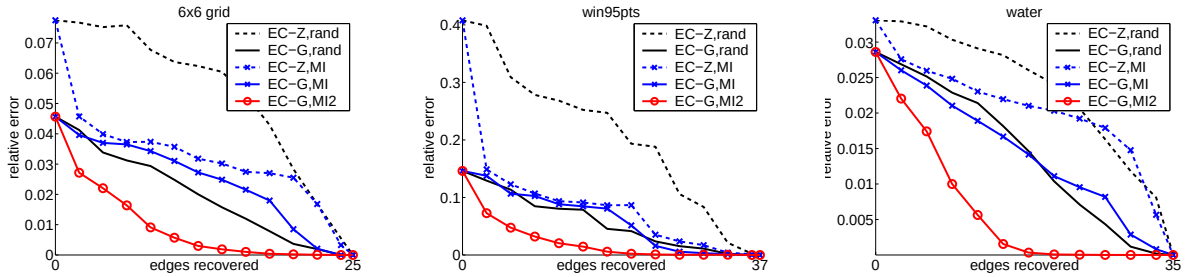


Figure 5: EC-Z versus EC-G, and edge recovery.

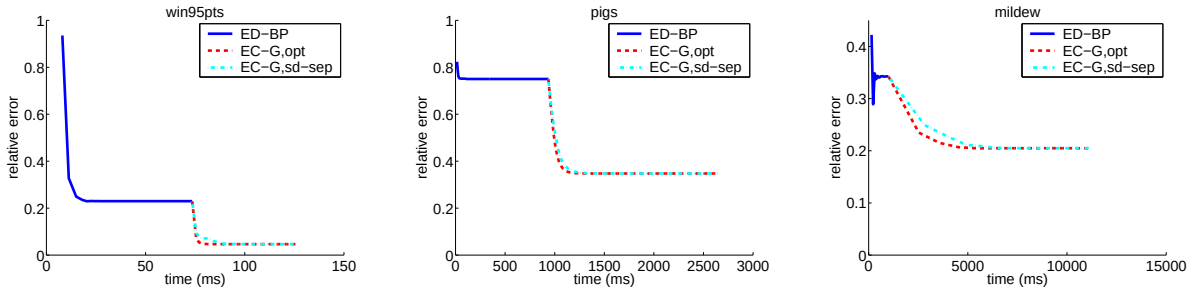


Figure 6: Time to parametrize by ED-BP, and compute EC-G corrections.

ingraph free energy approximations (Aji & McEliece, 2001; Dechter, Kask, & Mateescu, 2002); see (Choi & Darwiche, 2006a) for the connection to iterative joingraph propagation.

The EC-G correction can also take the form of another free energy approximation. Note first that when multiple equivalence edges are deleted, we can compute the partition function Z'_{ij} of a model $\mathcal{M}'_{ij}$ where the single edge (i,j) has been recovered (keeping edge parameters for all other edges fixed): $Z'_{ij} = Z' \cdot \frac{y_{ij}}{z_{ij}}$. Therefore, we have that:

$$Z' \cdot \frac{y}{z} = Z' \cdot \prod_{(i,j)\in\mathcal{E}^\star} \frac{y_{ij}}{z_{ij}} = Z' \cdot \prod_{(i,j)\in\mathcal{E}^\star} \frac{Z'_{ij}}{Z'}.$$

This yields a (dual) energy of the form $-\log(Z' \cdot \frac{y}{z}) = (n-1)\log Z' - \sum_{(i,j)\in\mathcal{E}^\star} \log Z'_{ij}$, where n is the number of equivalence edges (i,j) deleted. Whereas we fixed, somewhat arbitrarily, our edge parameters to satisfy Equation 1, we could in principle seek edge parameters optimizing the above free energy directly, giving rise to EP and GBP free energy approximations with higher-order structure (Welling, Minka, & Teh, 2005). On the other hand, edge recovery heuristics for EC-G could possibly serve as a heuristic for identifying improved EP and GBP free energies, directly. This is a perspective that is currently being investigated.

While we are concerned mostly with IBP and the closely related Bethe free energy approximation, we expect that an edge-correction perspective may be useful in improving other reasoning algorithms, particularly those that can be formulated as exact inference in simplified models. These include, as we have shown here, IBP and some of its generalizations (Yedidia et al., 2005), but also numerous variational methods (Jordan, Ghahramani, Jaakkola, & Saul, 1999; Wiegerinck, 2000; Geiger, Meek, & Wexler, 2006) and their corresponding free energy approximations. Also related, is tree-reweighted belief propagation (TRW) (Wainwright, Jaakkola, & Willsky, 2005), which provides upper bounds on the log partition function, and can be thought of as a convexified form of the Bethe free energy. Mean field methods and its generalizations are another well-known class of approximations that provide lower bounds on the partition function (e.g., Saul & Jordan, 1995; Jaakkola, 2001). Although the latter have been found to be useful, others have found that the Bethe free energy can often provide better quality approximations, (e.g., Weiss, 2001). Similarly, comparing EC-Z approximations and mean-field bounds derived from approximations with the same structure, we find that EC-Z, which does not guarantee bounds, offers better approximations.

8 CONCLUSION

We proposed an approach for approximating the partition function which is based on two steps: (1) computing the partition function of a simplified model which is obtained by deleting model edges, and (2) rectifying

the result by applying an edge-by-edge correction. The approach leads to an intuitive framework in which one can trade-off the quality of an approximation with the complexity of computing it through a simple process of edge recovery. We provided two concrete instantiations of the proposed framework by proposing two edge correction schemes with corresponding heuristics for edge recovery. The first of these instantiations captures the well known Bethe free energy approximation as a degenerate case. The second instantiation has been shown to lead to more accurate approximations, more so when edge recovery is targeted towards accurate correction. We further highlighted, in our experiments, how edge correction could be used as a conceptual tool to help identify situations where the Bethe approximation may be exact, or accurate. Finally, we suggested a notion of partial correction, that can improve on the Bethe approximation with only a modest amount of computational effort.

Acknowledgments

This work has been partially supported by Air Force grant #FA9550-05-1-0075 and by NSF grant #IIS-0713166.

A PROOFS

Note that Proposition 1 follows from Proposition 2.

Proof of Theorem 1 When a given model is a tree, the Bethe free energy is exact. We then consider the exact energy of a tree-structured $\mathcal{M}'$ where $F' = -\log Z'$. Our goal then is to show that $Z_\beta = Z' \cdot \frac{1}{z}$, or equivalently, $F' = F_\beta - \log z$.

Let $E[\,.\,]$ denote expectations and $H(.)$ denote entropies with respect to IBP beliefs, and equivalently, ED-BP marginals in $\mathcal{M}'$ (Choi & Darwiche, 2006a). First, note that $F_\beta = U_\beta - H_\beta$ where U_β is the Bethe *average* energy

$$U_\beta = -\sum_{(i,j)\in\mathcal{E}} E[\log \psi(X_i, X_j)\,] - \sum_{i\in\mathcal{V}} E[\log \psi(X_i)\,]$$

and where H_β is the Bethe *approximate* entropy

$$H_\beta = \sum_{(i,j)\in\mathcal{E}} H(X_i, X_j) - \sum_{i\in\mathcal{V}} (n_i - 1) H(X_i)$$

where n_i is the number of neighbors of node i in $\mathcal{M}$ (for details, see Yedidia et al., 2005).

It will be convenient to start with the case where every edge (i, j) in the unextended model is replaced with a chain $\{(i, i'), (i', j'), (j', j)\}$. We then delete all equivalence edges $(i, i'), (j', j) \in \mathcal{E}^\star$. Note that the resulting network $\mathcal{M}'$ has $n + 2m$ nodes: n nodes $i \in \mathcal{V}$, and 2 clone nodes i', j' for each of the m edges $(i', j') \in \mathcal{E}$.

The average energy U' and the entropy H' for $\mathcal{M}'$ is

$$\begin{aligned} U' = &-\sum_{(i',j')\in\mathcal{E}} E[\log \psi(X_i, X_j)\,] - \sum_{i\in\mathcal{V}} E[\log \psi(X_i)\,] \\ &- \sum_{(i,i')\in\mathcal{E}^\star} E[\log \theta(X_i)\theta(X'_i)\,] \\ H' = &\sum_{(i',j')\in\mathcal{E}^\star} H(X_i, X_j) + \sum_{i\in\mathcal{V}} H(X_i). \end{aligned}$$

Since $\theta(x_i)\theta(x_j) = z_{ij} Pr(x_i)$ (see Equation 2), we have

$$E[\log \theta(X_i)\theta(X'_i)\,] = \log z_{ij} - H(X_i). \qquad (5)$$

We can show through further manipulations that

$$\sum_{(i,i')\in\mathcal{E}^\star} E[\log \theta(X_i)\theta(X'_i)\,] = \log z - \sum_{i\in\mathcal{V}} n_i H(X_i).$$

After substituting into U'_β, and some rearrangement:

$$F' = U' - H' = U_\beta - H_\beta - \log z = F_\beta - \log z$$

as desired. To show this correspondence continues to hold for any tree-structured $\mathcal{M}'$, we note first that IBP beliefs continue to be node and edge marginals for any tree-structured ED-BP approximation $\mathcal{M}'$. Next, when we recover an edge into a tree approximation that yields another tree approximation, we lose an expectation over edge parameters (Equation 5). The corresponding node entropy $H(X_i)$ that is lost in the average energy U' is canceled out by a node entropy gained in the entropy H'. Finally, the term $\log z_{ij}$ that is lost is no longer needed in the correction factor z after recovery. Thus, we can recover edges into our fully disconnected approximation, and conclude that $F' = F_\beta - \log z$ continues to hold for any tree-structured approximation $\mathcal{M}'$. □

Proof of Proposition 2 In an extended network $\mathcal{M}$ with equivalence edge (i, j) and potential $\phi(x_i, x_j)$:

$$\begin{aligned} Z &= \sum_{x_i=x_j} \frac{\partial Z}{\partial \phi(x_i, x_j)} = \sum_{x_i=x_j} \frac{\partial^2 Z'}{\partial\theta(x_i)\partial\theta(x_j)} \\ &= \sum_{x_i=x_j} \frac{Z' Pr'(x_i, x_j)}{\theta(x_i)\theta(x_j)} = \sum_{x_i=x_j} \frac{Z' Pr'(x_i, x_j)}{z_{ij} Pr'(x_j)} \\ &= \frac{Z'}{z_{ij}} \sum_{x_i=x_j} Pr'(x_i \mid x_j) \end{aligned}$$

which is simply $Z' \cdot \frac{y_{ij}}{z_{ij}}$. Note that the fourth equality follows from Equation 2. □

Proof of Proposition 3 In an extended network $\mathcal{M}$ with equivalence edges (i, j) and (s, t) and edge potentials $\phi(x_i, x_j)$ and $\phi(x_s, x_t)$:

$$\begin{aligned}
Z &= \sum_{\substack{x_i=x_j \\ x_s=x_t}} \frac{\partial^2 Z}{\partial\phi(x_i,x_j)\partial\phi(x_s,x_t)} \\
&= \sum_{\substack{x_i=x_j \\ x_s=x_t}} \frac{\partial^4 Z'}{\partial\theta(x_i)\partial\theta(x_j)\partial\theta(x_s)\partial\theta(x_t)} \\
&= \sum_{\substack{x_i=x_j \\ x_s=x_t}} \frac{Z' Pr'(x_i,x_j,x_s,x_t)}{\theta(x_i)\theta(x_j)\theta(x_s)\theta(x_t)} \\
&= \sum_{\substack{x_i=x_j \\ x_s=x_t}} \frac{Z' Pr'(x_i,x_j,x_s,x_t)}{z_{ij} Pr'(x_j) z_{st} Pr'(x_t)} \quad \text{by Eq. 2} \\
&= \frac{Z'}{z_{ij} z_{st}} \sum_{\substack{x_i=x_j \\ x_s=x_t}} \frac{Pr'(x_i,x_j) Pr'(x_s,x_t)}{Pr'(x_j) Pr'(x_t)} \\
&= \frac{Z'}{z_{ij} z_{st}} \sum_{x_i=x_j} Pr'(x_i|x_j) \sum_{x_s=x_t} Pr'(x_s|x_t)
\end{aligned}$$

which is simply $Z' \cdot \frac{y_{ij}}{z_{ij}} \frac{y_{st}}{z_{st}}$. □

References

Aji, S. M., & McEliece, R. J. (2001). The generalized distributive law and free energy minimization. In *Proceedings of the 39th Allerton Conference on Communication, Control and Computing*, pp. 672–681.

Choi, A., & Darwiche, A. (2006a). An edge deletion semantics for belief propagation and its practical impact on approximation quality. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pp. 1107–1114.

Choi, A., & Darwiche, A. (2006b). A variational approach for approximating Bayesian networks by edge deletion. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 80–89.

Choi, A., & Darwiche, A. (2008). Many-pairs mutual information for adding structure to belief propagation approximations. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*. To appear.

Darwiche, A. (2003). A differential approach to inference in Bayesian networks. *Journal of the ACM*, *50*(3), 280–305.

Dechter, R., Kask, K., & Mateescu, R. (2002). Iterative join-graph propagation. In *UAI*, pp. 128–136.

Geiger, D., Meek, C., & Wexler, Y. (2006). A variational inference procedure allowing internal structure for overlapping clusters and deterministic constraints. *J. Artif. Intell. Res. (JAIR)*, *27*, 1–23.

Jaakkola, T. (2001). Tutorial on variational approximation methods. In Saad, D., & Opper, M. (Eds.), *Advanced Mean Field Methods*. MIT Press.

Jordan, M. I., Ghahramani, Z., Jaakkola, T., & Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, *37*(2), 183–233.

Park, J., & Darwiche, A. (2004). A differential semantics for jointree algorithms. *Artificial Intelligence*, *156*, 197–216.

Saul, L. K., & Jordan, M. I. (1995). Exploiting tractable substructures in intractable networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 486–492.

Wainwright, M. J., Jaakkola, T., & Willsky, A. S. (2003). Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, *49*(5), 1120–1146.

Wainwright, M. J., Jaakkola, T., & Willsky, A. S. (2005). MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, *51*(11), 3697–3717.

Weiss, Y. (2001). Comparing the mean field method and belief propagation for approximate inference in MRFs. In Saad, D., & Opper, M. (Eds.), *Advanced Mean Field Methods*. MIT Press.

Welling, M., Minka, T. P., & Teh, Y. W. (2005). Structured region graphs: morphing EP into GBP. In *UAI*, pp. 609–614.

Wiegerinck, W. (2000). Variational approximations between mean field theory and the junction tree algorithm. In *UAI*, pp. 626–633.

Yedidia, J., Freeman, W., & Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, *51*(7), 2282–2312.

Multi-View Learning in the Presence of View Disagreement

C. Mario Christoudias
UC Berkeley EECS & ICSI
MIT CSAIL

Raquel Urtasun
UC Berkeley EECS & ICSI
MIT CSAIL

Trevor Darrell
UC Berkeley EECS & ICSI
MIT CSAIL

Abstract

Traditional multi-view learning approaches suffer in the presence of view disagreement, i.e., when samples in each view do not belong to the same class due to view corruption, occlusion or other noise processes. In this paper we present a multi-view learning approach that uses a conditional entropy criterion to detect view disagreement. Once detected, samples with view disagreement are filtered and standard multi-view learning methods can be successfully applied to the remaining samples. Experimental evaluation on synthetic and audio-visual databases demonstrates that the detection and filtering of view disagreement considerably increases the performance of traditional multi-view learning approaches.

1 Introduction

Many problems in machine learning involve datasets that are naturally comprised of multiple views, e.g, web pages can be classified from their content or the content of the pages that point to them, an object can be categorized from either its color or shape. In a multi-modal setting, multiple views can be defined from separate sensory modalities, e.g., a person's agreement can be classified from their speech utterance or head gesture. Approaches to *multi-view learning* [1, 3, 5, 7, 12, 15, 17, 22] exploit multiple redundant views to effectively learn from unlabeled data by mutually training a set of classifiers defined in each view[1]. Multi-view learning can be advantageous when compared to learning with only a single view [3, 4, 12], especially when the weaknesses of one view complement the strengths of the other.

A common assumption in multi-view learning is that the samples from each view always belong to the same class. In realistic settings, datasets are often corrupted by noise. Multi-view learning approaches have difficulty dealing with noisy observations, especially when each view is corrupted by an independent noise process. For example, in multi-sensory datasets it is common that an observation in one view is corrupted while the corresponding observations in other views remain unaffected (e.g., the sensor is temporarily in an erroneous condition before returning back to normal behavior). Indeed, if the corruption is severe, the class can no longer be reliably inferred from the corrupted sample.

These corrupted samples can be seen as belonging to a "neutral" or "background" class that co-occur with uncorrupted observations in other views. The view corruption problem is thus a source of *view disagreement*, i.e., the samples from each view do not always belong to the same class but sometimes belong to an additional background class as a result of view corruption or noise. In this paper we present a method for performing multi-view learning in the presence of view disagreement caused by view corruption. Our approach treats each view as corrupted by a structured noise process and detects view disagreement by exploiting the joint view statistics using a conditional entropy measure.

We are particularly interested in inferring multi-modal semantics from weakly supervised audio-visual speech and gesture data. In audio-visual problems view disagreement often arises as a result of temporary view occlusion, or uni-modal expression (e.g., when expressing agreement a person may say 'yes' without head nodding).

The underlying assumption of our approach is that *foreground* samples can co-occur with samples of the

[1]Note that the views are redundant in that each class can be inferred from both views separately. In the idealized setting each view would be conditionally independent given the class label (e.g., see [3]).

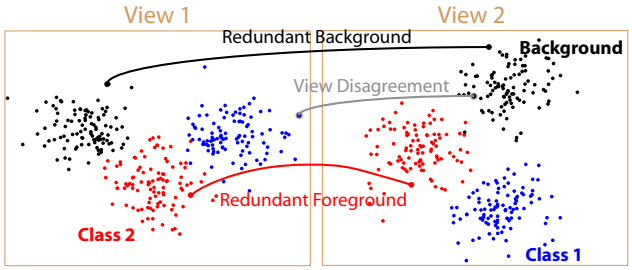


Figure 1: Synthetic two-view problem with normally distributed classes, two foreground and one background. Each view is 2-D; the two foreground classes are shown in red and blue. Corrupted samples form a separate background class (black samples) that co-occur with uncorrupted samples. For each point in the left view there is a corresponding point in the right view. Three point correspondences are shown: a redundant foreground sample, a redundant background sample and a sample with view disagreement where view 1 observed an instance of class 1, but view 2 for that sample was actually an observation of the background class. View disagreement occurs when one view is occluded and is incorrectly paired with background. Multi-view learning with these pairings leads to corrupted foreground class models.

same class or background, whereas background samples can co-occur with samples from any class, a reasonable assumption for many audio-visual problems. We define new multi-view bootstrapping approaches that use conditional entropy in a pre-filtering step to reliably learn in the presence of view disagreement. Experimental evaluation on audio-visual data demonstrates that the detection and filtering of view disagreement enables multi-view learning to succeed despite large amounts of view disagreement.

The remainder of this paper is organized as follows. In the next section, a discussion of multi-view learning approaches and view disagreement is provided. Our conditional entropy based criterion for detecting view disagreement is then outlined in Section 3 and our multi-view bootstrapping approach is presented in Section 4. Experimental results are provided in Section 5. A discussion of related methods and connections between our work and other statistical techniques is given in Section 6. Finally, in Section 7 we provide a summary and discuss future work.

2 Multi-View Learning

Several approaches to multi-view learning have been proposed in the machine learning literature [1, 3, 5, 12, 15, 17, 22]. In their seminal work, Blum and Mitchell [3] introduced co-training which bootstraps a set of classifiers from high confidence labels. Nigam and Ghani [15] presented a co-EM algorithm that uses soft label assignment with EM to bootstrap classifiers from multiple views. Collins and Singer [5] proposed a co-boost approach that optimizes an objective that explicitly maximizes the agreement between each classifier. Similarly, Sindhwani et. al. [17] defined a co-regularization method that learns a multi-view classifier from partially labeled data using a view consensus-based regularization term. More recently, Yu et. al. [22] presented a Bayesian co-training framework that defines a multi-view kernel for semi-supervised learning with Gaussian Processes.

Although there exists a wide variety of multi-view learning algorithms, they all function on the common underlying principle of view agreement. More formally, let $\mathbf{x}_k = (x_k^1, ..., x_k^V)$ be a multi-view sample with V views, and let $f_i : x^i \rightarrow \mathcal{Y}$ be the classifier that we seek in each view. Multi-view learning techniques train a set of classifiers $\{f_i\}$ by maximizing their consensus on the unlabeled data, $\mathbf{x}_k \in U$, for example by minimizing the L_2 norm [17],

$$\min \sum_{\mathbf{x_k} \in U} \sum_{i \neq j} \| f_i(x_k^i) - f_j(x_k^j) \|_2^2 \qquad (1)$$

The minimization in Eq. (1) is only applicable to multi-view learning problems for which the views are *sufficient* for classification, i.e., that classification can be performed from either view alone. In practice, however, it is often difficult to define views that are fully sufficient. Previous methods for overcoming insufficiency have addressed the case where both views are necessary for classification [5, 2, 17]. These methods formulate multi-view learning as a global optimization problem that explicitly maximizes the consensus between views. Although these approaches allow for views with partial insufficiency, they still assume that each view is largely sufficient. In the presence of significant view disagreement these approaches would in general diverge and perform poorly.

In this paper we identify and address a new form of insufficiency inherent to many real-world datasets, caused by samples where each view potentially belongs to a different class, e.g., as a result of view corruption. We refer to this form of insufficiency as the *view disagreement* problem. The view disagreement problem is distinct from the forms of view insufficiency that have been addressed in the literature—previous methods for overcoming insufficiency have addressed the case where both views are necessary for classification [1, 5, 14, 17], but not the case where the samples from each view potentially belong to different classes.

The problem of view disagreement exists in many real-

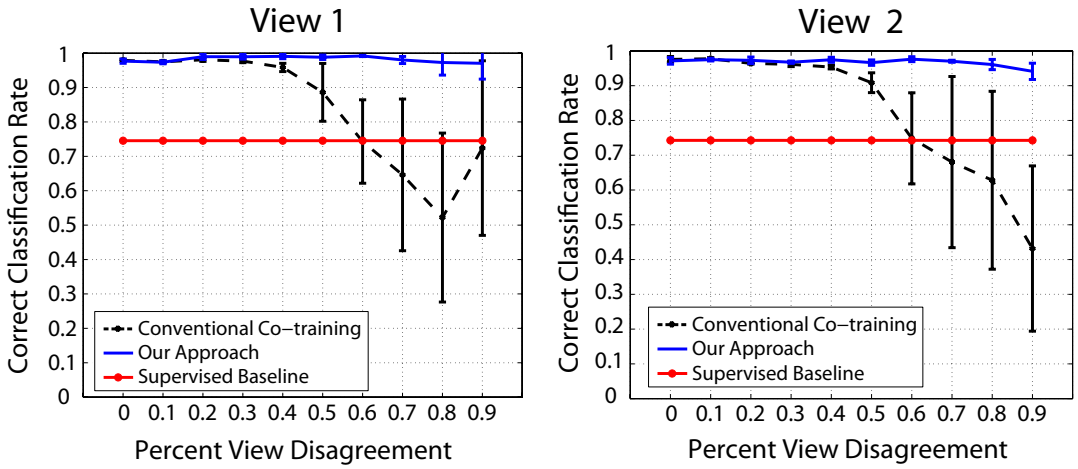


Figure 2: Multi-view learning for synthetic two-view example with varying amounts of view disagreement. Average performance is shown computed over 10 random splits of the training data into labeled and unlabeled sets; the error bars indicate ± 1 std. devation. Our approach exhibits good performance at all view disagreement levels while conventional co-training begins to diverge for percent disagreement greater than 40%.

world datasets. In user agreement recognition from head gesture and speech [4], people often say 'yes' without head nodding and vice versa, and/or the subject can also become temporary occluded in either the audio or visual modalities by other speakers or objects in the scene. In semantic concept detection from text and video [21], it is possible for the text to describe a different event than what is being displayed in the video. Another example is web-page classification from page and hyper-link content [3], where the hyperlinks can often point to extraneous web-pages not relevant to the classification task.

We illustrate the problem of view disagreement in multi-view learning with a toy example containing two views of two foreground classes and one background class. The samples of each class are drawn from Gaussian distributions with unit variance (see Figure 1). Figure 2 shows the degradation in performance of conventional co-training [3] for varying amounts of view disagreement. Here, co-training is evaluated using a left out test set and by randomly splitting the training set into labeled and unlabeled datasets. We report average performance across 10 random splits of the training data. As shown in Figure 2 co-training performs poorly when subject to significant amounts of view disagreement ($\geq 40\%$).

In what follows, we present a method for detecting view disagreement using a measure of conditional view entropy and demonstrate that when used as a prefiltering step, our approach enables multi-view learning to succeed despite large amounts of view disagreement.

3 Detection and Filtering of View Disagreement

We consider an occlusion process where an additional class models background. We assume that this background class can co-occur with any of the $n+1$ classes in the other views[2], and that the n foreground classes only co-occur with samples that belong to the same class or background, as is common in audio-visual datasets [4].

In this paper we propose a conditional entropy criterium for detecting samples with view disagreement. We further assume that background co-occurs with more than one foreground class; this is a reasonable assumption for many types of background (e.g., audio silence). In what follows, we treat each view x^i, $i = 1, ..., V$ as a random variable and detect view disagreement by examining the joint statistics of the different views. The entropy $H(x)$ of a random variable is a measure of its uncertainty [6]. Similarly, the conditional entropy $H(x|y)$ is a measure of the uncertainty in x given that we have observed y. In the multi-view setting, the conditional entropy between views, $H(x^i|x^j)$, can be used as a measure of agreement that indicates whether the views of a sample belong to the same class or event. In what follows, we call $H(x^i|x^j)$ the *conditional view entropy*.

Under our assumptions we expect the conditional view entropy to be larger when conditioning on background

[2]Note that background samples can co-occur with the any of the n foreground classes plus background.

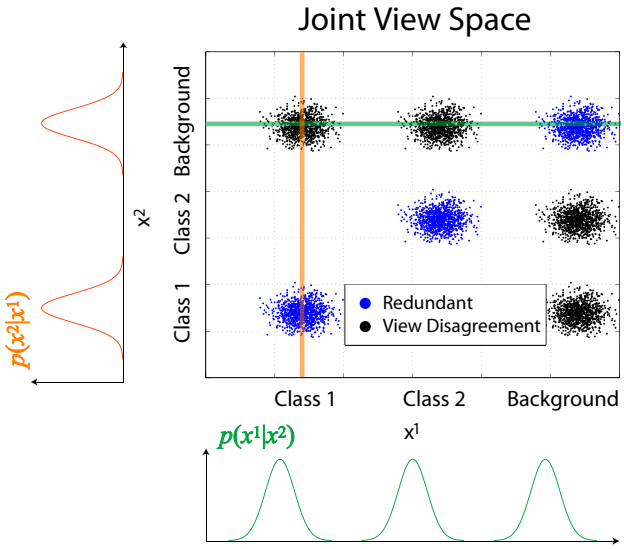


Figure 3: View disagreement caused by view corruption. The joint view space of a two-view problem with view disagreement is displayed. Redundant samples are highlighted in blue and samples with view disagreement in black. The conditional distributions for a sample with view disagreement are shown. The conditional distribution resulting from conditioning on background exhibits more peaks and therefore has a higher uncertainty than when conditioning on foreground.

compared to foreground. Thus, we have $\forall p = 1, ...n$,

$$H(x^i|x_k^j \in C^{n+1}) > H(x^i|x_l^j \in C^p) \tag{2}$$

where C^i is the set of examples belonging to class i. A notional example of view corruption is illustrated in Figure 3. This example contains two, 1-D views of two foreground classes and one background class. As before, the samples of each class are drawn from a normal distribution with unit variance. The conditional view distributions of a multi-view sample with view disagreement is displayed. Note that the uncertainty of view i when conditioning on view j has greater uncertainty when view j is background.

We delineate foreground from background samples by thresholding the conditional view entropy. In particular, we define the threshold in each view using the mean conditional entropy computed over the unlabeled data. More formally, let (x_k^i, x_k^j) be two different views of a multi-view sample $\mathbf{x}_k = (x_k^1, ..., x_k^V)$. We define an indicator function, $m(\cdot)$, that operates over view pairs (x^i, x^j) and that is 1 if the conditional entropy of x^i conditioned on x_k^j is below the mean conditional entropy,

$$m(x^i, x_k^j) = \begin{cases} 1, & H(x^i|x_k^j) < \bar{H}_{ij} \\ 0, & \text{otherwise} \end{cases}, \tag{3}$$

with

$$H(x^i|x_k^j) = -\sum_{x^i \in U^i} p(x^i|x_k^j) \log p(x^i|x_k^j), \tag{4}$$

where U^i is the ith view of the unlabeled dataset, and $\bar{H}_{ij}$ is the mean conditional entropy,

$$\bar{H}_{ij} = \frac{1}{M} \sum_{\mathbf{x}_k \in U} H(x^i|x_k^j), \tag{5}$$

where M is the number of samples in U. $m(x^i, x^j)$ is used to detect whether x^j belongs to foreground since under our model foreground samples have a low conditional view entropy.

A sample $\mathbf{x}_k$ is a *redundant foreground* sample if it satisfies

$$\prod_{i=1}^{V} \prod_{j \neq i} m(x^i, x_k^j) = 1. \tag{6}$$

Similarly, $\mathbf{x}_k$ is a *redundant background* sample if it satisfies

$$\sum_{i=1}^{V} \sum_{j \neq i} m(x^i, x_k^j) = 0. \tag{7}$$

A multi-view sample x_k is in *view disagreement* if it is neither a redundant foreground nor a redundant background sample.

Definition 1. Two views (x_k^i, x_k^j) of a multi-view sample $\mathbf{x}_k$ are in view disagreement if

$$m(x^i, x_k^j) \oplus m(x^j, x_k^i) = 1 \tag{8}$$

where $\oplus$ is the logical xor operator that has the property that $a \oplus b$ is 1 iff $a \neq b$ and 0 otherwise.

Eq. (8) defines our conditional entropy criterion for view disagreement detection between pairs of views of a multi-view sample.

In practice, we estimate the conditional probability of Eq. (4) as

$$p(x^i|x_k^j) = \frac{f(x^i, x_k^j)}{\sum_{x^i \in U^i} f(x^i, x_k^j)} \tag{9}$$

where $f(\mathbf{x})$ is a multivariate kernel density estimator[3]. In our experiments, the bandwidth of f is set using automatic bandwidth selection techniques [16].

[3]Note our approach is agnostic to the choice of probability model and more sophisticated conditional probability models can be used, such as [20], that perform better in high dimensional input spaces.

Algorithm 1 Multi-View Bootstrapping in the Presence of View Disagreement

Given classifiers f_i and labeled seed sets S_i, $i = 1, ..., V$, unlabeled dataset U and parameters N and T:
Set $t = 1$.
repeat
  for $i = 1, ..., V$ **do**
    Train f_i on S_i
    Evaluate f_i on U^i
    Sort U in decreasing order by f_i confidence
    for each $\mathbf{x}_k \in U$, $k = 1, ..., N$ **do**
      for $j \neq i$ **do**
        if $\neg(m(x^i, x^j_k) \oplus m(x^j, x^i_k))$ **then**
          $U^j = U^j \backslash \{x^j_k\}$
          $S^j = S^j \cup \{x^j_k\}$
        end if
      end for
      $U^i = U^i \backslash \{x^i_k\}$
      $S^i = S^i \cup \{x^i_k\}$
    end for
  end for
  Set $t = t + 1$.
until $|U| = \emptyset$ or $t = T$

4 Multi-view Bootstrapping in the Presence of View Disagreement

In this section we present a new multi-view bootstrapping algorithm that uses the conditional entropy measure of Eq. (8) in a pre-filtering step to learn from multi-view datasets with view disagreement.

Multi-view bootstrapping techniques, e.g., co-training, mutually train a set of classifiers, f_i, $i = 1, ..., V$, on an unlabeled dataset U by iteratively evaluating each classifier and re-training from confidently classified samples. The classifiers are initialized from a small set of labeled examples typically referred to as the *seed set*, S. During bootstrapping, confidently classified samples in each view are used to label corresponding samples in the other views. It has been shown that multi-view bootstrapping is advantageous to self-training with only a single view [4].

We extend multi-view bootstrapping to function in the presence of view disagreement. A separate labeled set, S_i, is maintained for each view during bootstrapping and the conditional entropy measure of Eq. (8) is checked before labeling samples in the other views from labels in the current view. The parameters to the algorithm are N, the number of samples labeled by each classifier during each iteration of bootstrapping, and T the maximum number of multi-view bootstrapping iterations. The resulting algorithm self-trains each classifier using all of the unlabeled examples, and only enforces a consensus on the samples with view agreement (see Algorithm 1).

Algorithm 2 Cross-Modality Bootstrapping in the Presence of View Disagreement

Given existing classifier f_1 and initial classifier f_2, unlabeled dataset U and parameter N:

Initialization:
Sort U in decreasing order by f_1 confidence
Define $L = \{(y_k, x^2_k)\}$, $k = 1, ..., N$

Bootstrapping:
Set $S = \emptyset$
for each $(y_k, x^2) \in L$ **do**
  if $\neg(m(y, x^2_k) \oplus m(x^2, y_k))$ **then**
    $S = S \cup \{(y_k, x^2_k)\}$
    $L = L \backslash \{(y_k, x^2_k)\}$
  end if
end for
Train f_2 on S.

Figure 2 displays the result of multi-view bootstrapping for the toy example of Figure 1 using $N = 6$ and T was set such that all the unlabeled data was used. With our method, multi-view learning is able to proceed successfully despite the presence of severe view disagreement and is able to learn accurate classifiers in each view even when presented with datasets that contain up-to 90% view disagreement.

In audio-visual problems it is commonly the case that there is an imbalance between the classification difficulty in each view. In such cases, an accurate classifier can be learned in the weaker view using an unsupervised learning method that bootstraps from labels output by the classifier in the other view. Here, the class labels output by the classifier in the stronger view can be used as input to the conditional entropy measure as they provide a more structured input than the original input signal.

The resulting cross-modality bootstrapping algorithm trains a classifier f_2 in the second view from an existing classifier f_1 in the first view on a two-view unlabeled dataset U. The algorithm proceeds as follows. First f_1 is evaluated on U and the N most confidently classified examples are moved from U to the labeled set L. The conditional entropy measure is then evaluated over each label, sample pair $(y, x^2) \in L$, where $y = f_1(x^1)$. The final classifier f_2 results from training on the the samples in L that are detected as redundant foreground or redundant background (see Algorithm 2).

5 Experimental Evaluation

We evaluate the performance of multi-view bootstrapping techniques on the task of audio-visual user agreement recognition from speech and head gesture. Al-

though users often use redundant expression of agreement, it is frequently the case that they say 'yes' without head gesturing and viceversa. View disagreement can also be caused by noisy acoustic environments (e.g., a crowded room), temporary visual occlusions by other objects in the scene, or if the subject is temporarily out of the camera's field of view.

To evaluate our approach we used a dataset of 15 subjects interacting with an avatar in a conversational dialog task [4]. The interactions included portions where each subject answered a set of yes/no questions using head gesture and speech. The head gesture consisted of head nods and shakes and the speech data of 'yes' and 'no' utterances. In our experiments, we simulate view disagreement in the visual domain using both no motion (i.e., random noise) and real background head motion samples from non-response portions of the interaction. Similarly, background in the audio is simulated as babble noise.

The visual features consist of 3-D head rotation velocities output by a 6-D head tracker [13]. For each subject, we post-process these observations by computing a 32 sample windowed Fast Fourier Transform (FFT) separately over each dimension, with a time window of 1 second corresponding to the expected length of a head gesture. The resulting sequence of FFT observations is then segmented using the avatar's transcript which marks the beginning and end of each user response.

The FFT spectra of each user response were amplitude normalized and blurred in space and time to remove variability to location, duration and rate of head motion. Principle Components Analysis (PCA) was then performed over the vector space resulting from flattening the FFT spectra corresponding to each response into a single vector. The resulting 3-D PCA space captured over 90% of the variance and was computed over the unlabeled samples of the training set.

The audio features consist of 13-D Mel Frequency Cepstral Coefficients (MFCCs) sampled at 100Hz over the segmented audio sequences corresponding to each user response, obtained from the avatar's transcript. The audio sequences were then converted into single frame observations using the technique of [11]. In this representation, an audio sequence is divided into portions and an average MFCC vector is computed over each portion. In our experiments, we used proportions equal to $(0.3, 0.4, 0.3)$. The concatenated averages along with first derivatives and log duration define a 61-D observation vector. To reduce the dimensionality of this space, PCA was applied retaining 98% of the variance which resulted in a 9-D, single-frame audio observation space.

In our experiments we use correct classification rate as the evaluation metric, defined as:

$$\text{CCR} = \frac{\#\text{ of examples correctly classified}}{\text{total }\#\text{ of examples}} \tag{10}$$

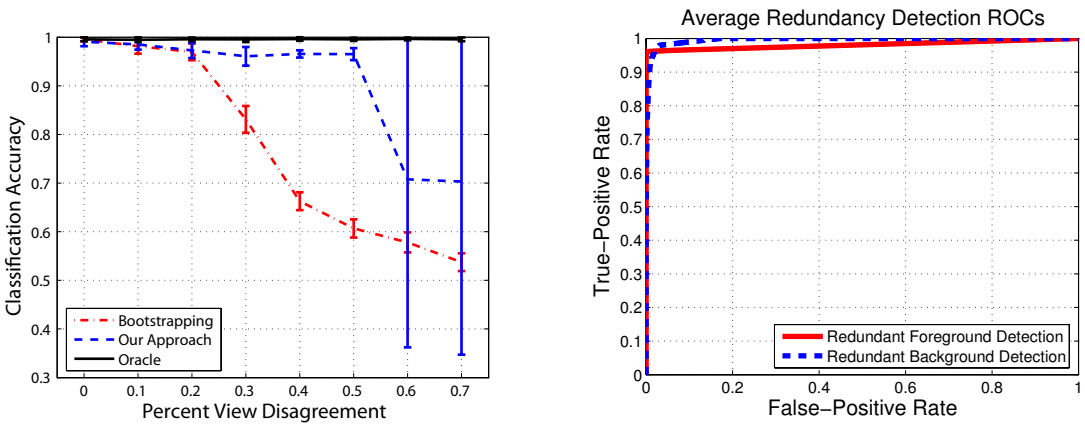
We used Bayes classifiers for audio and visual gesture recognition defined as $p(y|x) = \frac{p(x|y)}{\sum_y p(x|y)}$, where $p(x|y)$ is Gaussian. Specifically, Bayes classifiers for $p(y|x^a)$ and $p(y|x^v)$ are bootstrapped from semi-supervised audio-visual data; x^a and x^v correspond to audio and visual observations respectively.

5.1 Cross-Modality Bootstrapping

First, we evaluate our cross-modality bootstrapping approach. For this task, we are interested in performing semi-supervised learning of visual head gesture by bootstrapping from labels in the speech (e.g., those output by an off-the-shelf speech recognizer). We simulated view disagreement by randomly replacing observations in the visual modality with background sequences, and replacing labels in the audio with the background label. Redundant background was also added such that there were an equal number of redundant background samples as there were redundant foreground samples per class.

We first show results using a "no motion" visual background modeled as zero mean Gaussian noise in the 3-D head rotational velocity space with $\sigma = 0.1$. Figure 4 displays the result of evaluating the performance of multi-view bootstrapping (Algorithm 2) with varying amounts of view disagreement. Performance is shown averaged over 5 random splits of the data into 10 train and 5 test subjects. At small amounts of view disagreement ($\leq$ 20%) conventional bootstrapping and our approach exhibit similar good performance. When the view disagreement is small the error can be viewed as classification noise in the audio. For larger amounts of view disagreement (up to 50%), conventional multi-view bootstrapping diverges and our algorithm still succeeds in learning an accurate head gesture recognizer from the audio-visual data. For $>$ 50% view disagreement, our approach begins to degrade and exhibits a large variance in performance. This high variability can be a result of poor bandwidth selection, or a poor choice of threshold. We plan to investigate alternative methods for modeling the conditional probability and more sophisticated threshold selection techniques as part of future work.

Figure 4(b) displays average receiver-operator curves (ROCs) for redundant foreground and background class detection that result from varying the entropy threshold of the conditional entropy measure. The mean conditional entropy defines a point on these



(a) (b)

Figure 4: Bootstrapping a user agreement visual classifier from audio. (a) Performance is shown averaged over random splits of the data into 10 train and 5 test subjects over varying amounts of simulated view disagreement using a no motion background class; error bars indicate ± 1 std. deviation. Unlike conventional bootstrapping, our approach is able to cope with up-to 50% view disagreement. (b) Average view disagreement detection ROCs are also shown for redundant foreground and background detection. Our approach effectively detects view disagreement.

curves. As illustrated by the figure, overall our approach does fairly well in detecting view disagreement.

Next, we consider a more realistic occluding visual background class generated from randomly selecting head motion sequences from non-response portions of each user interaction. In contrast to the "no motion" class considered above, these segments contain miscellaneous head motion in addition to no motion.

Our view disagreement detection approach (Algorithm 2) performs equally well in the presence of the more challenging real background as is shown in Figure 5. As before, conventional bootstrapping performs poorly in the presence of view disagreement. In contrast, our approach is able to successfully learn a visual classifier in the presence of significant view disagreement (up to 50%).

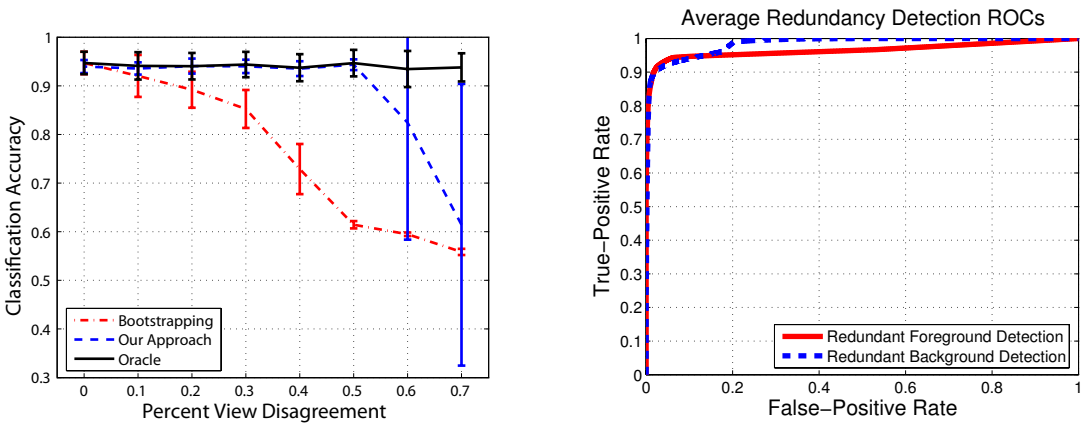
5.2 Multi-View Bootstrapping

We evaluated the performance of multi-view bootstrapping (Algorithm 1) for the task of semi-supervised learning of audio-visual user agreement classifiers from speech and head gesture. Figure 6 displays the result of audio-visual co-training for varying amounts of view disagreement. Performance is shown averaged over 5 random splits of the data into 10 train and 5 test subjects and over 10 random splits of the training data into labeled seed set and unlabeled training set, with 15 labeled samples, 5 per class. Conventional co-training and our approach were then evaluated using $N = 6$ and $T = 100$. We chose N such that the classes are balanced.

For this problem, the initial visual classifier trained from the seed set is much more accurate than the initial audio classifier that performs near chance. The goal of co-training is to learn accurate classifiers in both the audio and visual modalities. Note, that in contrast to cross-modality bootstrapping, this is done without any a priori knowledge as to which modality is more reliable. For small amounts of view disagreement ($\geq 20\%$), both conventional co-training and our approach (Algorithm 1) are able to exploit the strong performance in the visual modality to train an accurate classifier in the audio. For larger amounts of view disagreement, conventional co-training begins to diverge and at the 70% view disagreement level is not able to improve over the supervised baseline in both the audio and visual modalities. In contrast, our approach reliably learns accurate audio-visual classifiers across all view disagreement levels.

6 Discussion

Recently, Ando and Zhang [1] presented a multi-view learning approach that instead of assuming a consensus over classification functions assume that the views share the same low dimensional manifold. This has the advantage that it can cope with insufficient views where classification cannot be performed from either view alone. Still, their approach defines a consensus between views, and therefore assumes that the samples in each view are of the same class. View disagreement will violate this assumption and we expect their method to degrade as multi-view bootstrapping.



(a) (b)

Figure 5: Bootstrapping a user agreement visual classifier from audio with real visual background. Performance is shown averaged over random splits of the data into 10 train and 5 test subjects over varying amounts of simulated view disagreement; error bars indicate ± 1 std. deviation. The conventional bootstrapping baseline performs poorly in the presence of view disagreement. In contrast, our approach is able to (a) successfully learn a visual classifier and (b) classify views in the presence of significant view disagreement (up to 50%).

Our approach treats each view as corrupted by a structured noise process and detects view disagreement by exploiting the joint view statistics. An alternative method to coping with view disagreement is to treat each view as belonging to a stochastic process and use a measure such as mutual information to test for view dependency [19, 18]. In [18], Siracusa and Fisher use hypothesis testing with a hidden factorization Markov model to infer dependency between audio-visual streams. It would be interesting to apply such techniques for performing multi-view learning despite view disagreement, which we leave as part of future work.

Our work bears similarity to co-clustering approaches which use co-occurrence statistics to perform multi-view clustering [10, 9, 8]. These techniques, however, do not explore the relationship between co-occurrence and view sufficiency and would suffer in the presence of view disagreement since the occluding background would potentially cause foreground clusters to collapse into a single cluster.

We demonstrated our view disagreement detection and filtering approach for multi-view bootstrapping techniques (e.g., [3, 15, 4]). However, our algorithm is generally applicable to any multi-view learning method and we believe it will be straightforward to adapt it for use with other approaches (e.g., [1, 5, 17]). Multi-view learning methods either implicity or explicitly maximize the consensus between views to learn from unlabeled data; view disagreement adversatively affects multi-view learning techniques since they encourage agreement between views.

In our experiments, our approach performs well on a realistic dataset with noisy observations. The success of our approach on this dataset is predicated on the fact that foreground and background classes exhibit distinct co-occurrence patterns, which our algorithm exploits to reliably detect view disagreement.

7 Conclusions and Future Work

In this paper we have identified a new multi-view learning problem, view disagreement, inherent to many real-word multi-view datasets. We presented a multi-view learning framework for performing semi-supervised learning from multi-view datasets in the presence of view disagreement and demonstrated that a conditional entropy criterion was able to detect view disagreement caused by view corruption or noise. As shown in our experiments, for the task of audio-visual user agreement our method was able to successfully perform multi-view learning even in the presence of gross view disagreement ($50 - 70\%$). Interesting avenues for future work include the investigation of alternative entropy threshold selection techniques, the use of alternative probability models for computing conditional entropy and modeling redundancy between non-stationary stochastic processes using measures such as mutual information.

References

[1] R. K. Ando and T. Zhang. Two-view feature generation model for semi-supervised learning. In *ICML*, 2007.

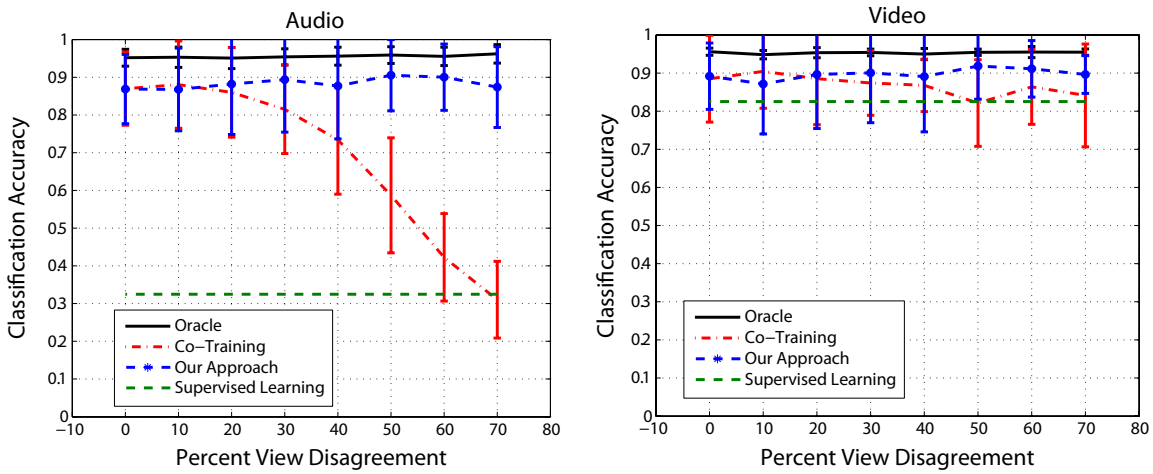


Figure 6: Multi-view bootstrapping of audio-visual user agreement classifiers. Performance of (a) audio and (b) video is displayed averaged over 5 random splits of the data into 10 train and 5 test subjects and over 10 random splits of each training set into labeled seed set and unlabeled dataset; error bars show ±1 std. deviation. Conventional co-training performs poorly in the presence of significant view disagreement. In contrast, our approach performs well across all view disagreement levels.

[2] S. Bickel and T. Scheffer. Estimation of mixture models using co-em. In *Proceedings of the European Conference on Machine Learning*, 2005.

[3] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.

[4] C. M. Christoudias, K. Saenko, L.-P. Morency, and T. Darrell. Co-adaptation of audio-visual speech and gesture classifiers. In *ICMI*, November 2006.

[5] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

[6] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley and Sons, New York, second edtion, 2006 edition, 1991.

[7] S. Dasgupta, M. Littman, and D. Mcallester. PAC generalization bounds for co-training. In *NIPS*, 2001.

[8] V. R. de Sa. Spectral clustring with two views. In *Proceedings of the European Conference on Machine Learning*, 2005.

[9] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Knowledge Discovery and Data Mining*, pages 269–274, 2001.

[10] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD*, pages 89–98, 2003.

[11] A. Halberstadt. *Heterogeneous Acoustic Measurements and Multiple Classiers for Speech Recognition*. PhD thesis, MIT, 1998.

[12] S. M. Kakade and D. P. Foster. Multi-view regression via canonical correlation analysis. In *COLT*, 2007.

[13] L.-P. Morency, A. Rahimi, and T. Darrell. Adaptive view-based appearance model. In *CVPR*, 2003.

[14] I. Muslea, S. Minton, and C. A. Knoblock. Adaptive view validation: A first step towards automatic view detection. In *ICML*, 2002.

[15] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of cotraining. In *Workshop on Information and Knowledge Management*, 2000.

[16] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986.

[17] V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularization approach to semi-supervised learning with multiple views. In *International Conference on Machine Learning*, 2005.

[18] M. R. Siracusa and J. W. Fisher III. Dynamic dependency tests: Analysis and applications to multi-modal data association. In *AIStats*, 2007.

[19] M.R. Siracusa, K. Tieu, A. Ihler, J. Fisher III, and A.S. Willsky. Estimating dependency and significance for high-dimensional data. In *ICASSP*, 2005.

[20] R. Urtasun and T. Darrell. Local probabilistic regression for activity-independent human pose inference. In *CVPR*, 2008.

[21] R. Yan and M. Naphade. Semi-supervised cross feature learning for semantic concept detection in videos. In *CVPR*, June 2005.

[22] S. Yu, B. Krishnapuram, R. Rosales, H. Steck, and R. B. Rao. Bayesian co-training. In *NIPS*, 2007.

Bounds on the Bethe Free Energy for Gaussian Networks

Botond Cseke
Faculty of Science
Radboud University Nijmegen
Toernooiveld 1, 6525 ED
Nijmegen, The Netherlands

Tom Heskes
Faculty of Science
Radboud University Nijmegen
Toernooiveld 1, 6525 ED
Nijmegen, The Netherlands

Abstract

We address the problem of computing approximate marginals in Gaussian probabilistic models by using mean field and fractional Bethe approximations. As an extension of Welling and Teh (2001), we define the Gaussian fractional Bethe free energy in terms of the moment parameters of the approximate marginals and derive an upper and lower bound for it. We give necessary conditions for the Gaussian fractional Bethe free energies to be bounded from below. It turns out that the bounding condition is the same as the pairwise normalizability condition derived by Malioutov et al. (2006) as a sufficient condition for the convergence of the message passing algorithm. By giving a counterexample, we disprove the conjecture in Welling and Teh (2001): even when the Bethe free energy is not bounded from below, it can possess a local minimum to which the minimization algorithms can converge.

1 Introduction

Calculating marginal probabilities of a set of variables given some observations is one of the major tasks of probabilistic inference. In the case of Gaussian models, the computation of the marginal probabilities has a computational complexity that scales cubically with the number of variables, while for models with discrete variables, it often leads to intractable computations. Computations can be made faster or tractable by using approximate inference methods like mean field approximation (e.g., Jaakkola, 2000) and Bethe approximation (e.g., Yedidia et al., 2000). These methods were developed mainly for discrete probabilistic graphical models, but they are applicable in Gaussian models as well. However, there are important differences in their behavior for the discrete and Gaussian cases. For example, while the error function of the Bethe approximation—also called Bethe free energy—in discrete models is bounded from below, in Gaussian models this is not always the case (see Welling and Teh, 2001).

The study of the Bethe free energy of Gaussian models is also motivated by their importance for the study of conditional Gaussian models. Conditional Gaussian or hybrid graphical models, such as switching Kalman filters (e.g., Zoeter and Heskes, 2005), combine both discrete and Gaussian variables. Approximate inference in these models can be carried out by expectation propagation (e.g., Minka, 2004, 2005). Expectation propagation can be viewed as a generalization of the Bethe approximation where marginalization constraints are replaced by expectation constraints (e.g., Heskes et al., 2005). Therefore, studying the properties of the Bethe free energy can reveal some of the convergence properties of expectation propagation. In order to understand the properties of the Bethe free energy of hybrid models a good understanding of the two special cases of discrete and Gaussian models is needed. While the properties of the Bethe free energy of discrete models have been studied extensively in the last decade and are well understood (e.g., Yedidia et al., 2000; Heskes, 2003; Wainwright et al., 2003), the properties of the Gaussian Bethe free energy have been studied much less.

The message passing algorithm is a well established method for finding the stationary points of the Bethe free energy (e.g., Pearl, 1988; Yedidia et al., 2000; Heskes, 2003). It works by locally updating the approximate marginals and has been successfully applied in both discrete (e.g., Murphy et al., 1999; Wainwright et al., 2003) and Gaussian models (e.g., Weiss and Freeman, 2001; Rusmevichientong and Roy, 2001; Malioutov et al., 2006). The problem of finding sufficient conditions for the convergence of message passing in Gaussian networks has been successfully addressed

by many authors. Using the computation tree approach, Weiss and Freeman (2001) proved that message passing converges whenever the information—inverse covariance—matrix of the probability distribution is diagonally dominant[1]. With the help of an analogy between message passing and walk–sum analysis, Malioutov et al. (2006) derived the stronger condition of pairwise normalizability[2]. A different approach was taken by Welling and Teh (2001), who directly minimized—with regard to the parameters of approximate marginals—the Bethe free energy, conjecturing that Gaussian message passing converges if and only if the free energy is bounded from below. Their experiments showed that message passing and direct minimization either converge to the same solution or both fail to converge.

Following Welling and Teh (2001), instead of analyzing the message passing updates, we turn our attention to the properties of the Bethe free energy expressed as a function of the moment parameters of approximate marginals. We derive a lower bound for the Gaussian fractional Bethe free energies and give necessary conditions for them to be bounded from below.

2 Background

The probability distribution of a Gaussian undirected probabilistic graphical model—also known as Gaussian Markov random field—is usually defined in terms of canonical parameters, namely $p(\mathbf{x}) \propto \exp\left\{\mathbf{h}^T\mathbf{x} - \frac{1}{2}\mathbf{x}^T\mathbf{J}\mathbf{x}\right\}$—with $\mathbf{J}$ symmetric and positive definite. Such canonical parameterizations often result from Bayesian computations, for example from a model with a Gaussian likelihood and Gaussian prior. The calculation of marginals requires matrix inversions, that is they can be computed as $p(\mathbf{x}_I) \propto \exp\left\{(\mathbf{h}_I - \mathbf{J}_{I,R}\mathbf{J}_{R,R}^{-1}\mathbf{h}_R)^T\mathbf{x}_I - \frac{1}{2}\mathbf{x}_I^T\mathbf{J}_{I,R}\mathbf{J}_{R,R}^{-1}\mathbf{J}_{R,I}\mathbf{x}_I\right\}$, for any $I \cap R = \emptyset$ with $I \cup R = \{1,\dots,N\}$—here, N is the number of variables in the model. In sparse models, the complexity of computations can be reduced to scale with the number of non-zero elements of $\mathbf{J}_{R,R}$, but computing marginals for several groups of variables can still be costly. Trading correctness for speed, one can opt for computing approximate marginals.

A popular method to approximate marginals is approximating p with a distribution q having a form that makes marginals easy to identify. The most common quantity to measure the difference between two probability distributions is the Kullback-Leibler divergence $\mathrm{D}\left[q||p\right]$. It is often used (e.g., Jaakkola, 2000) to characterize the quality of the approximation and formulate the computation of approximate marginals as the optimization problem

$$q(\mathbf{x}) = \underset{q\in\mathcal{F}}{\operatorname{argmin}} \int q(\mathbf{x}) \log\left[\frac{q(\mathbf{x})}{p(\mathbf{x})}\right] d\mathbf{x}. \qquad (1)$$

Here, $\mathcal{F}$ is the set of distributions with the above mentioned form. Since it is not symmetric, the Kullback-Leibler divergence is not a distance, but $\mathrm{D}\left[q||p\right] \geq 0$ for any proper q and p, $\mathrm{D}\left[q||p\right] = 0$ if and only if $p = q$ and it is convex in both q and p.

A family $\mathcal{F}$ of densities possessing a form that makes marginals easy to identify is the family of distributions that factorize as $q(\mathbf{x}) = \prod_k q_k(x_k)$. In other words, in problem (1) we approximate p with a distribution that has independent variables. An approximation q of this type is called mean field approximation (e.g., Jaakkola, 2000). Writing out in detail the right hand side of (1) one gets

$$F_{MF}(\{q_k\}) = -\int \log p(\mathbf{x}) \prod_k q_k(x_k) d\mathbf{x} + \sum_k \int q_k(x_k) \log q_k(x_k) dx_k.$$

Using notation $q_k(x_k) = N(x_k|m_i,\sigma_i^2), \mathbf{m} = (m_1,\dots,m_N)^T$ and $\boldsymbol{\sigma} = (\sigma_1,\dots,\sigma_N)^T$, this simplifies to

$$F_{MF}\left(\mathbf{m},\boldsymbol{\sigma}\right) = -\left\{\mathbf{h}^T\mathbf{m} - \frac{1}{2}\mathbf{m}^T\mathbf{J}\mathbf{m} - \frac{1}{2}\sum_k J_{kk}\sigma_k^2\right\} - \sum_k \log(\sigma_k) + C_{MF}, \qquad (2)$$

where C_{MF} is an irrelevant constant. Although $\mathrm{D}\left[\prod_k q_k||p\right]$ might not be convex in $(q_1,\dots,q_N)$, one can easily check that F_{MF} is convex in its variables $\mathbf{m}$ and $\boldsymbol{\sigma}$ and its minimum is obtained for $\mathbf{m} = \mathbf{J}^{-1}\mathbf{h}$ and $\boldsymbol{\sigma} = 1/\sqrt{\mathrm{diag}\left(\mathbf{J}\right)}$. Since $[\mathbf{J}^{-1}]_{kk} = 1/(J_{kk} - \mathbf{J}_{k,\setminus k}^T\left[\mathbf{J}_{\setminus k,\setminus k}\right]^{-1}\mathbf{J}_{\setminus k,k})$, one can easily see that the mean field approximation underestimates variances. Note that the mean field approximation computes a solution in which the means are exact, but the variances are computed as if there were no interactions between the variables, namely as if the matrix $\mathbf{J}$ were diagonal, thus giving poor estimates of the variances.

In order to improve the estimates for variances, one has to choose approximating distributions q that are

[1] The matrix $\mathbf{A}$ is diagonally dominant if $|A_{ii}| > \sum_{j\neq i}|A_{ij}|$ for all i.

[2] Following Malioutov et al. (2006), we call a Gaussian distribution pairwise normalizable if it can be factorized into a product of normalizable "pair" factors, that is $p(x_1,\dots,x_n) = \prod_{ij}\Psi_{ij}(x_i,x_j)$ such that all Ψ_{ij}-s are normalizable.

able to capture dependencies between the variables in p. It can be verified that any distribution in which the dependencies form a tree graph can be written in the form

$$p(\mathbf{x}) = \prod_{n(i,j)} \frac{p(x_i, x_j)}{p(x_i)p(x_j)} \prod_k p(x_k),$$

where i and j run through all the connections or edges $n(i,j)$ of the tree and k runs through $\{1, \ldots, N\}$. Although in most cases the undirected graph generated by the connections in $\mathbf{J}$ is not a tree, based on the "tree intuition" one can construct q from one and two variable marginals as

$$q(\mathbf{x}) \propto \prod_{n(i,j)} \frac{q_{ij}(x_i, x_j)}{q_i(x_i)q_j(x_j)} \prod_k q_k(x_k) \tag{3}$$

and constrain the functions q_{ij} and q_k to be marginally consistent and normalize to 1, that is $\int q_{ij}(x_i, x_j)dx_j = q_i(x_i)$ for any i and j and $\int q_k(x_k)dx_k = 1$ for any k. An approximation of the form (3) together with the constraints on q_{ij}–s and q_k–s is called a Bethe approximation. Denoting the family of such functions by $\mathcal{F}_B$, by choosing $q_{ij}(x_i, x_j) = q_i(x_i)q_j(x_j)$ one can easily check that $\mathcal{F}_{MF} \subset \mathcal{F}_B$, thus $\mathcal{F}_B$ is non-empty. Assuming that the approximate marginals are correct and q normalizes to 1 and then substituting (3) into (1), we get an approximation of the Kullback–Leibler divergence in (1) called the Bethe free energy. Since the interactions between the variables in p are pairwise, we can factorize p as $p(\mathbf{x}) \propto \prod_{n(i,j)} \Psi_{i,j}(x_i, x_j)$, and express the Bethe free energy as

$$\begin{aligned} F_B(\{q_{ij},q_k\}) = &- \sum_{n(i,j)} \int q_{ij}(\mathbf{x}_{i,j}) \log \Psi_{ij}(\mathbf{x}_{i,j}) d\mathbf{x}_{i,j} \\ &+ \sum_{n(i,j)} \int q_{ij}(\mathbf{x}_{i,j}) \log \left[\frac{q_{ij}(\mathbf{x}_{i,j})}{q_i(x_i)q_j(x_j)} \right] d\mathbf{x}_{i,j} \\ &+ \sum_k \int q_k(x_k) \log q_k(x_k) dx_k. \end{aligned} \tag{4}$$

Yedidia, Freeman, and Weiss (2000) showed that the fixed point iteration for finding the constrained minima of the function in (4), boils down to the message passing algorithm of Pearl (1988). The algorithm is derived from the Karush–Kuhn–Tucker conditions of the constrained minimization. As it was mentioned in the introduction, in case of Gaussian models this algorithm does not always converge, and the reason for this appears to be that the approximate marginals may get indefinite or negative definite covariance matrices. Welling and Teh (2001) pointed out that this can be due to the unboundedness of the Bethe free energy.

Since F_{MF} is convex and bounded and the Bethe free energy could be unbounded, it seems plausible to analyze the fractional Bethe free energy

$$\begin{aligned} F_{\boldsymbol{\alpha}}(\{q_{ij}, q_k\}) = &- \sum_{n(i,j)} \int q_{ij}(\mathbf{x}_{i,j}) \log \Psi_{ij}(\mathbf{x}_{i,j}) d\mathbf{x}_{i,j} \\ &+ \sum_{n(i,j)} \frac{1}{\alpha_{ij}} \int q_{ij}(\mathbf{x}_{i,j}) \log \left[\frac{q_{ij}(\mathbf{x}_{i,j})}{q_i(x_i)q_j(x_j)} \right] d\mathbf{x}_{i,j} \\ &+ \sum_k \int q_k(x_k) \log q_k(x_k) dx_k \end{aligned} \tag{5}$$

introduced by Wiegerinck and Heskes (2003). Here, $\boldsymbol{\alpha}$ denotes the set of variables $\{\alpha_{ij}\}$. They showed that the fractional Bethe free energy "interpolates" between the mean field and the Bethe approximation. That is for $\alpha_{ij} = 1$ we get the Bethe free energy, while in the case when all α_{ij}–s tend to 0 the mutual information between variables x_i and x_j is highly penalized, therefore, (5) enforces solutions close to the mean field solution. They also showed that the fractional message passing algorithm derived from (5) can be interpreted as Pearl's message passing algorithm with the difference that instead of computing local marginals—like in Pearl's algorithm—one computes local α_{ij}–marginals[3]. The local α_{ij}–marginals correspond to "true" local marginals when $\alpha_{ij} = 1$ and to local mean field approximations when $\alpha_{ij} = 0$.

Power expectation propagation by Minka (2004) is an approximate inference method that uses local approximations with α–divergences. It turns out that in case of Gaussian models power expectation propagation—with a fully factorized approximating distribution—boils down to the same message passing algorithm as the one derived from (5) and the appropriate constraints.

Starting from the idea of creating a convex upper bound of the Bethe free energy when p and q are exponential distributions, Wainwright et al. (2003) derived a form of (5) where the α_{ij}–s are chosen such that it is convex in $(\{q_{ij}\}, \{q_k\})$. Thus, they derived a form of the fractional Bethe free energy that has a unique global minimum.

3 Main results

In this section we analyze the parametric form of (5). Setting all α_{ij} values equal, we show that the fractional Gaussian Bethe free energy is a non-increasing function of α. By letting α tend to infinity, we obtain a lower bound for the free energies. It turns out

[3]Here, we define the α–marginals of a distribution p as $\operatorname{argmin}_{\{q_k\}} D_\alpha \left[p \parallel \prod_k q_k \right]$, where D_α is the α–divergence.

that the condition for the lower bound to be bounded from below is the same as the pairwise normalizability condition.

Conforming to Malioutov et al. (2006) and without loss of generality, we work with the "normalized" information matrix, that is we use $\mathbf{J} = \mathbf{I}+\mathbf{R}$ where $\mathrm{diag}(\mathbf{R}) = \mathbf{0}$. We define $|\mathbf{R}|$ as the matrix formed by the absolute values of $\mathbf{R}$'s elements. Following Welling and Teh (2001), we use $q_{ij}(\mathbf{x}_{i,j}) = N(\mathbf{x}_{i,j}|\mathbf{m}_{ij}, \mathbf{\Sigma}_{ij})$ and $q_k(x_k) = N(x_k|m_k, \sigma_k^2)$, where $\mathbf{m} = (m_1, \ldots, m_N)^T$, $\mathbf{m}_{i,j} = (m_i, m_j)^T$ and $\mathbf{\Sigma}_{ij} = \left[\sigma_i^2,\ \sigma_{ij};\ \sigma_{ij},\ \sigma_j^2\right]$, and thus we embed the marginalization and normalization constraints into the parameterization. The matrix formed by diagonal elements σ_k^2 and off-diagonal elements σ_{ij} is denoted by $\mathbf{\Sigma}$ and the vector of standard deviations by $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_N)^T$. Substituting q_{ij} and q_k into (5) one gets

$$\begin{aligned} F_{\boldsymbol{\alpha}}\left(\mathbf{m}, \mathbf{\Sigma}\right) = &- \left\{ \mathbf{h}^T\mathbf{m} - \frac{1}{2}\mathbf{m}^T\mathbf{J}\mathbf{m} - \frac{1}{2}\mathrm{Tr}(\mathbf{J}^T\mathbf{\Sigma}) \right\} \\ &- \frac{1}{2}\sum_{n(i,j)} \frac{1}{\alpha_{ij}} \log\left(1 - \frac{\sigma_{ij}^2}{\sigma_i^2\sigma_j^2}\right) \\ &- \sum_k \log(\sigma_k) + C_{\boldsymbol{\alpha}} \end{aligned} \tag{6}$$

where $C_{\boldsymbol{\alpha}}$ is an irrelevant constant. Note that the variables $\mathbf{m}$ and $\mathbf{\Sigma}$ are independent, hence the minimizations of $F_{\boldsymbol{\alpha}}\left(\mathbf{m}, \mathbf{\Sigma}\right)$ with regard to $\mathbf{m}$ and $\mathbf{\Sigma}$ can be carried out independently.

Property 1. *$F_{\boldsymbol{\alpha}}\left(\mathbf{m}, \mathbf{\Sigma}\right)$ is convex and bounded in $(\mathbf{m}, \{\sigma_{ij}\}_{i\neq j})$ and at any stationary point we have*

$$\begin{aligned} \mathbf{m}^* &= \mathbf{J}^{-1}\mathbf{h} \qquad (7) \\ {\sigma_{ij}}^* &= -\mathrm{sign}(R_{ij})\frac{\left(1 + (2\alpha_{ij}R_{ij})^2\sigma_i^2\sigma_j^2\right)^{1/2} - 1}{2\alpha_{ij}|R_{ij}|}. \end{aligned}$$

Proof: By definition $\mathbf{J}$ is positive definite, therefore, the quadratic term in $\mathbf{m}$ is convex and bounded. The variables $\mathbf{m}$ and $\mathbf{\Sigma}$ are independent and the minimum with regard to $\mathbf{m}$ is achieved at $\mathbf{m}^* = \mathbf{J}^{-1}\mathbf{h}$.

One can check that the second order derivative of $F_{\boldsymbol{\alpha}}(\mathbf{m}, \mathbf{\Sigma})$ with regard to σ_{ij} is non-negative and the first order derivative has only one solution when $-\sigma_i\sigma_j \leq \sigma_{ij} \leq \sigma_i\sigma_j$ (Welling and Teh, 2001). Since the variables σ_{ij} are independent, one can conclude that $F_{\boldsymbol{\alpha}}(\mathbf{m}, \mathbf{\Sigma})$ is convex in $\{\sigma_{ij}\}$. From the independence of $\mathbf{m}$ and $\mathbf{\Sigma}$, it follows that $F_{\boldsymbol{\alpha}}$ is convex in $(\mathbf{m}, \{\sigma_{ij}\})$. ■

Since $\mathbf{\Sigma}_{ij}$ is constrained to be a covariance matrix, we have $\sigma_{ij}^2 \leq \sigma_i^2\sigma_j^2$, thus the first logarithmic term in (6) is negative. As a consequence, by setting all α_{ij}–s equal, we get,

$$F_{\alpha_1}(\mathbf{m}, \mathbf{\Sigma}) \geq F_{\alpha_2}(\mathbf{m}, \mathbf{\Sigma}) \quad \text{for any } \alpha_1 \leq \alpha_2.$$

This leads to the following property.

Property 2. *With $\alpha_{ij} = \alpha$, F_α is a non-increasing function of α.*

Using Property 1 and substituting σ_{ij}^* into $F_{\boldsymbol{\alpha}}$ we define the constrained function

$$\begin{aligned} F_{\boldsymbol{\alpha}}^c(\mathbf{m}, \boldsymbol{\sigma}) = &\frac{1}{2}\mathbf{m}\mathbf{J}^{-1}\mathbf{m} - \mathbf{h}^T\mathbf{m} + \frac{1}{2}\sum_k \sigma_k^2 \\ &- \frac{1}{2}\sum_{n(i,j)} \frac{1}{\alpha_{ij}} \left(\left(1 + (2\alpha_{ij}R_{ij})^2\sigma_i^2\sigma_j^2\right)^{1/2} - 1 \right) \\ &- \frac{1}{2}\sum_{n(i,j)} \frac{1}{\alpha_{ij}} \log\left(2\frac{\left(1 + (2\alpha_{ij}R_{ij})^2\sigma_i^2{\sigma_j}^2\right)^{1/2} - 1}{(2\alpha_{ij}R_{ij})^2\sigma_i^2\sigma_j^2} \right) \\ &- \sum_k \log(\sigma_k) + C_{\boldsymbol{\alpha}}^c \end{aligned} \tag{8}$$

where $C_{\boldsymbol{\alpha}}^c$ is an irrelevant constant. From Property 2, it follows that when choosing $\alpha_{ij} = \alpha$, the function in (8) is a non-increasing function of α. It then makes sense to take $\alpha \to \infty$ and verify whether we can get a lower bound for (8).

Lemma *For any $\boldsymbol{\sigma} > 0$, $0 \leq \alpha_1 \leq 1$ and $\alpha_2 \geq 1$ the following inequalities hold.*

$$\begin{aligned} F_{MF}(\mathbf{m}, \boldsymbol{\sigma}) &\geq F_{\alpha_1}^c(\mathbf{m}, \boldsymbol{\sigma}) \geq F_B(\mathbf{m}, \{\sigma_{ij}^*\}, \boldsymbol{\sigma}) \\ F_B(\mathbf{m}, \{\sigma_{ij}^*\}, \boldsymbol{\sigma}) &\geq F_{\alpha_2}^c(\mathbf{m}, \boldsymbol{\sigma}) \ldots \\ \ldots &\geq F_{MF}(\mathbf{m}, \boldsymbol{\sigma}) - \frac{1}{2}\boldsymbol{\sigma}^T|\mathbf{R}|\boldsymbol{\sigma} \end{aligned}$$

Moreover, they are tight, that is

$$\lim_{\alpha\to 0} F_\alpha(\mathbf{m}, \{\sigma_{ij}^*(\alpha)\}, \boldsymbol{\sigma}) = F_{MF}(\mathbf{m}, \boldsymbol{\sigma})$$

and

$$\lim_{\alpha\to\infty} F_\alpha(\mathbf{m}, \{\sigma_{ij}^*(\alpha)\}, \boldsymbol{\sigma}) = F_{MF}(\mathbf{m}, \boldsymbol{\sigma}) - \frac{1}{2}\boldsymbol{\sigma}^T|\mathbf{R}|\boldsymbol{\sigma}.$$

Proof: Since the Bethe free energy is the specific case of the fractional Bethe free energy for $\alpha = 1$, the inequalities on $F_B(\mathbf{m}, \{\sigma_{ij}^*(\alpha)\}, \boldsymbol{\sigma})$ follow from Property 2. Now, we show that the upper and lower bounds are tight. The function $(1 + x^2)^{1/2} - 1$ behaves as $\frac{1}{2}x^2$ in the neighborhood of 0, therefore,

$$\lim_{\alpha\to 0} \sigma_{ij}^*(\alpha) = 0$$

and

$$\lim_{\alpha\to 0} \frac{\log\left(1 - \frac{{\sigma_{ij}^*}^2(\alpha)}{\sigma_i^2\sigma_j^2}\right)}{\alpha} = -\frac{1}{\sigma_i^2\sigma_j^2}\lim_{\alpha\to 0}\frac{{\sigma_{ij}^*}^2(\alpha)}{\alpha} = 0,$$

showing that $F_{MF}(\mathbf{m}, \boldsymbol{\sigma})$ is a tight upper bound. As α tends to infinity, we have

$$\lim_{\alpha\to\infty} \frac{\left(1 + (2\alpha R_{ij})^2\sigma_i^2\sigma_j^2\right)^{1/2} - 1}{2\alpha} = |R_{ij}|\sigma_i\sigma_j$$

and

$$\lim_{\alpha\to\infty} \frac{1}{\alpha} \log\left(\frac{\left(1 + (2\alpha R_{ij})^2\sigma_i^2{\sigma_j}^2\right)^{1/2} - 1}{(2\alpha R_{ij})^2\sigma_i^2\sigma_j^2}\right) = 0,$$

yielding

$$\lim_{\alpha\to\infty} F_\alpha(\mathbf{m}, \{\sigma_{ij}^*(\alpha)\}, \boldsymbol{\sigma}) = F_{MF}(\mathbf{m}, \boldsymbol{\sigma}) - \frac{1}{2}\boldsymbol{\sigma}^T|\mathbf{R}|\boldsymbol{\sigma}.$$

■

Let $\lambda_{max}(|\mathbf{R}|)$ be the largest eigenvalue of $|\mathbf{R}|$. Analyzing the boundedness of the lower bound, we arrive at the following theorem.

Theorem *For the fractional Bethe free energy in (6) corresponding to a connected Gaussian network, the following statements hold*

(1) if $\lambda_{max}(|\mathbf{R}|) < 1$, then $F_{\boldsymbol{\alpha}}$ is bounded from below for all $\boldsymbol{\alpha} > \mathbf{0}$,

(2) if $\lambda_{max}(|\mathbf{R}|) > 1$, then $F_{\boldsymbol{\alpha}}$ is unbounded from below for all $\boldsymbol{\alpha} > \mathbf{0}$,

(3) if $\lambda_{max}(|\mathbf{R}|) = 1$ then $F_{\boldsymbol{\alpha}}$ is bounded from below if and only if $\frac{1}{2}\sum_i \sum_{n(i,j)} \frac{1}{\alpha_{ij}} \geq N$.

Proof: Since in $F_{\boldsymbol{\alpha}}$ there is no interaction between the parameters $\mathbf{m}$ and $\boldsymbol{\Sigma}$ and the term depending on $\mathbf{m}$ is bounded from below due to the positive definiteness of $\mathbf{J}$, we can simply neglect this term when analyzing the boundedness of $F_{\boldsymbol{\alpha}}$. Let us write out in detail the lower bound of the fractional Bethe free energies in the form

$$F_{MF}(\mathbf{m}, \boldsymbol{\sigma}) - \frac{1}{2}\boldsymbol{\sigma}^T|\mathbf{R}|\boldsymbol{\sigma} = \frac{1}{2}\mathbf{m}^T\mathbf{J}^{-1}\mathbf{m} - \mathbf{h}^T\mathbf{m} + \frac{1}{2}\boldsymbol{\sigma}^T\left(\mathbf{I} - |\mathbf{R}|\right)\boldsymbol{\sigma} - \sum_k \log(\sigma_k) + C, \qquad (9)$$

Statement (1): The condition $\lambda_{max}(|\mathbf{R}|) < 1$ implies that $\mathbf{I} - |\mathbf{R}|$ is positive definite. Now, $\log(x) \leq x - 1$, thus $\frac{1}{2}\boldsymbol{\sigma}^T(\mathbf{I} - |\mathbf{R}|)\boldsymbol{\sigma} - \mathbf{1}^T\log(\boldsymbol{\sigma}) \geq \frac{1}{2}\boldsymbol{\sigma}^T(\mathbf{I} - |\mathbf{R}|)\boldsymbol{\sigma} - \mathbf{1}^T\boldsymbol{\sigma} + N$. The latter is bounded from below and so it follows that (9) is bounded from below as well. According to the Lemma, boundedness of (9) implies that all fractional Bethe free energies are bounded from below.

Statement (2): Since we assumed that the Gaussian network is connected and undirected, it follows that $|\mathbf{R}|$ is irreducible (e.g., Horn and Johnson, 2005). According to the Frobenius-Perron theory of non-negative matrices (e.g., Horn and Johnson, 2005), the non-negative and irreducible matrix $|\mathbf{R}|$ has a simple maximal eigenvalue $\lambda_{max}(|\mathbf{R}|)$ and all elements of the eigenvector $\mathbf{u}_{max}$ corresponding to it are positive. Let us take the fractional Bethe free energy and analyze its behavior when $\boldsymbol{\sigma} = t\mathbf{u}_{max}$ and $t \to \infty$. Since for large values of t we have $\left(1 + (2\alpha_{ij}R_{ij})^2(u_{max}^i u_{max}^j)^2 t^4\right)^{1/2} \simeq 2\alpha_{ij}|R_{ij}|u_{max}^i u_{max}^j t^2$, the sum of the second and third term in (8) boils down to $(1 - \lambda_{max}(|\mathbf{R}|))t^2$ and this term dominates over the logarithmic ones as $t \to \infty$. As a result, the limit is independent of the choice of α_{ij} and it tends to $-\infty$ whenever $\lambda_{max}(|\mathbf{R}|) > 1$.
Statement (3): If $\lambda_{max}(|\mathbf{R}|) = 1$, then the only direction in which the quadratic term will not dominate is $\boldsymbol{\sigma} = t\mathbf{u}_{max}$. Therefore, we have to analyze the behavior of the logarithmic terms in (8) when $t \to \infty$. For large t-s these terms behave as $\left(\frac{1}{2}\sum_i \sum_{n(i,j)} \frac{1}{\alpha_{ij}} - N\right)\log(t)$. For this reason, the boundedness of $F_{\boldsymbol{\alpha}}^c$—and thus of $F_{\boldsymbol{\alpha}}$—depends on the condition in statement (3). ■

It was shown by Malioutov et al. (2006) that the condition $\lambda_{max}(|\mathbf{R}|) < 1$ is an equivalent condition of pairwise normalizability. Therefore, pairwise normalizability is not only a sufficient condition for the message passing algorithm to converge, but it is also a necessary condition for the fractional Gaussian Bethe free energies to be bounded.

Example In the case of models with K–regular adjacency matrix (non-zero entries of $\mathbf{R}$) and equal interaction weights $R_{ij} = r$, the maximal eigenvalue of $|\mathbf{R}|$ is $\lambda_{max}(|\mathbf{R}|) = Kr$ and the eigenvector corresponding to this eigenvalue is $\mathbf{1}$. (We define $\mathbf{1}$ as the vector that has all its elements equal to 1.) Verifying the stationary point conditions, it turns out that for some choice of r and α there exists a local minimum which is symmetrical, that is it lies in the direction $\mathbf{1}$. One can show that when the model is not pairwise normalizable ($Kr > 1$), the critical r below which the fractional Bethe free energy possesses this local minimum is $r_c(K, \alpha) = 1/2\sqrt{\alpha(K - \alpha)}$ and for any valid r the critical α below which the fractional Bethe free energies possesses this local minimum is $\alpha_c(K, r) = \frac{1}{2}K\left(1 - \sqrt{1 - 1/(Kr)^2}\right)$. These results are illustrated in Figure 1. (Note that for 2–regular graphs, all valid models are pairwise normalizable and possess a unique global minimum.) ■

For K–regular graphs convexity of the fractional Bethe

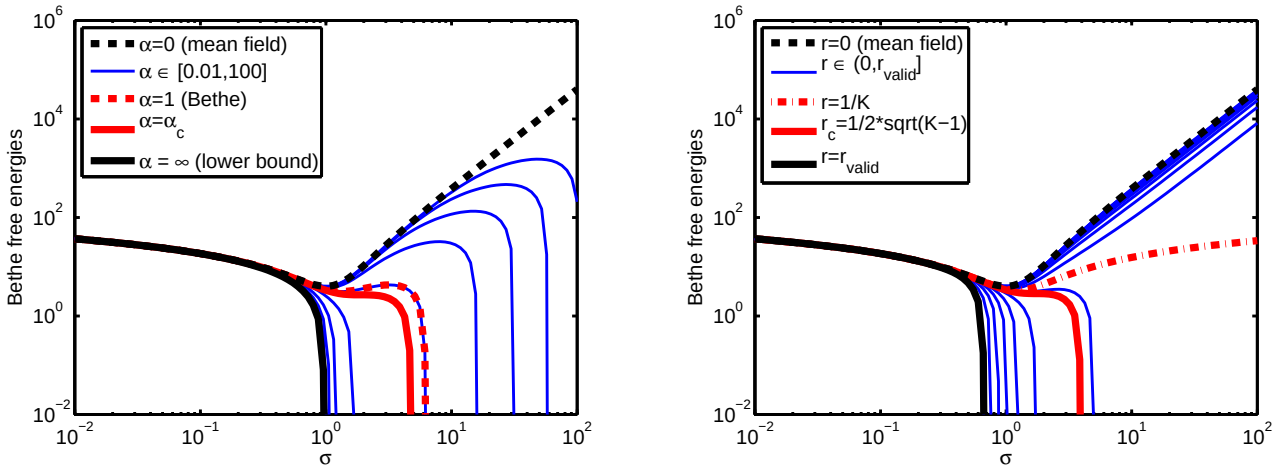


Figure 1: Visualizing critical parameters for a symmetric K-regular Gaussian model with $R_{ij} = r$. Plots in the left panel correspond to the constrained fractional Bethe free energies F_α^c in the direction $\mathbf{1}$ for an 8 node 4–regular Gaussian model with r=0.27 ($Kr > 1$) and varying α. Plots in the right panel correspond to the constrained Bethe free energies F_1^c in the direction $\mathbf{1}$ for an 8 node 4–regular Gaussian model with varying r. Here, r_{valid} is the supremum of r–s for which the model is valid—$\mathbf{J}$ is positive definite.

free energy in terms of $\{q_{ij}, q_k\}$ requires $\alpha \geq K$, a much stronger condition than $\alpha \geq \alpha_c(K, r)$. Thus, if we choose α sufficiently large such that the Bethe free energy is guaranteed to have a unique global minimum, this minimum is unbounded.

4 Experiments

We implemented both direct minimization and fractional message passing and analyzed their behavior for different values of $\lambda_{max}(|\mathbf{R}|)$. For reasons of simplicity we set all α_{ij} equal. The results are summarized in Figure 2. Note that there is a good correspondence between the behavior of the fractional Bethe free energies in the direction of the eigenvalue corresponding to $\lambda_{max}(|\mathbf{R}|)$ and the convergence of the Newton method. The Newton method was started from varying initial points. We experienced that when $\lambda_{max}(|\mathbf{R}|) > 1$ and setting the initial value to $t\mathbf{u}_{max}$, the algorithm did not converge for high values of t. This can be explained by the plots in Figure 2: for high values of t the initial point is not in the convergence region of the local minimum. For the fractional message passing algorithm we used two types of initialization: (1) when $\lambda_{max}(|\mathbf{R}|) < 1$ we set Ψ_{ij} such that they are all normalizable and set initial messages to 1, (2) when $\lambda_{max}(|\mathbf{R}|) \geq 1$, we used a symmetric partitioning of p into Ψ_{ij}-s —symmetric partitioning of the diagonal elements of $\mathbf{J}$—and set messages to identical values such that in the first step of the algorithm all approximate marginals become normalizable.

We experienced a behavior similar to that described by Welling and Teh (2001) for standard message passing, namely fractional message passing and direct minimization either both converge or both fail to converge. Our experiments in combination with the Theorem show that when $\lambda_{max}(|\mathbf{R}|) > 1$ standard message passing at best converges to a local minimum of the Bethe free energy. If standard message passing fails to converge, one can decrease α and search for a stationary point—preferably a local minimum—of the corresponding fractional free energy.

It can be seen from the results in the right panels of Figure 1 and 2, that when the model is no longer pairwise normalizable, the local minimum and not the unbounded global minimum is the natural continuation of the (bounded) global minimum for pairwise normalizable models. This explains why the quality of the approximation at the local minimum for models that are not pairwise normalizable is still comparable to that at the global minimum for models that are pairwise normalizable. Note that when the model is pairwise normalizable both the upper and lower bounds are convex in t. This may be obscured by the use of logarithmic axes.

5 Conclusions and future research

As we have seen, F_{MF} and $F_{MF} - \frac{1}{2}\boldsymbol{\sigma}^T|\mathbf{R}|\boldsymbol{\sigma}$ provide tight upper and lower bounds for the Gaussian fractional Bethe free energies. It turns out that pairwise normalizability (see Malioutov et al., 2006) is not only a sufficient condition for the message passing algorithm to converge, but it is also a necessary condition for the

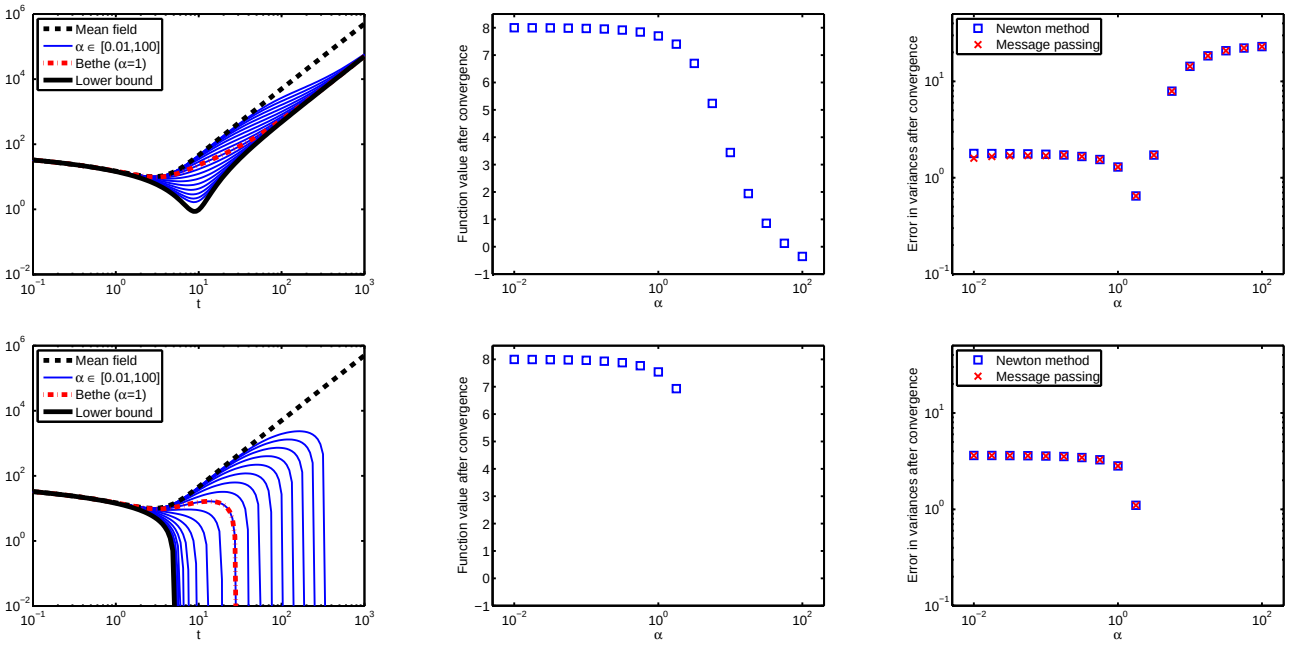


Figure 2: Left panels show the constrained fractional Bethe free energies of an 8 node Gaussian network in the direction of the eigenvector corresponding to $\lambda_{max}(|\mathbf{R}|)$ for $\lambda_{max}(|\mathbf{R}|) = 0.9$ (top) and $\lambda_{max}(|\mathbf{R}|) = 1.1$ (bottom). The thick lines are the functions F_{MF} (dashed), F_B (dashed dotted) and the lower bound $F_{MF} - \frac{1}{2}\sigma^T|\mathbf{R}|\sigma$ (continuous). The thin lines are the constrained α-fractional free energies F_α^c for $\alpha \in [10^{-2}, 10^2]$. Center panels show the final function values after the convergence of the Newton method. Right panels show the $||\cdot||_2$ error in approximation for the single node standard deviations $\boldsymbol{\sigma}$. Missing values indicate non-convergence.

Gaussian fractional Bethe free energies to be bounded from below.

If the model is pairwise normalizable, then the lower bound is bounded from below, therefore, both direct minimization and fractional message passing are converging for any choice of $\boldsymbol{\alpha} > \mathbf{0}$. Experiments show that regardless of the initialization, they converge to the same minimum. This suggests that in the pairwise normalizable case, fractional Bethe free energies possess a unique global minimum.

If the model is not pairwise normalizable, then none of the fractional Bethe free energies is bounded from below. However, experiments show that there is always a range of α values for which both direct minimization and fractional message passing converge. By decreasing α towards zero, one gets closer to the mean field energy and a local minimum to which the minimization algorithms can converge may appear.

As mentioned in Section 2, α_{ij}–s correspond to using local α_{ij} divergences when applying power expectation propagation with a fully factorized approximating distribution. Several papers on continuous models (e.g., Minka, 2004; Qi et al., 2005) report that when expectation propagation does not converge, applying power expectation propagation with $\alpha < 1$ helps to achieve convergence. In the case of the problem addressed in this paper this behavior can be explained by the observation that small α–s make local minima more likely to occur and thus prevents the covariance matrices from becoming indefinite or even non positive definite.

Wainwright et al. (2003) propose to convexify the Bethe free energy by choosing α_{ij}–s sufficiently large such that the fractional Bethe free energy has a unique global minimum. This strategy appears to fail for Gaussian models. Convexification makes the possibly useful local minima dissapear, leaving just the unbounded global minimum. In the case of the more general hybrid models the use of the convexification is still unclear.

Note that the example in the previous section disproves the conjecture in Welling and Teh (2001): even when the Bethe free energy is not bounded from below, it can possess a local minimum to which the minimization algorithms converge.

Our future goals are to find sufficient conditions for the Bethe free energy to possess local minima and to derive algorithms that are guaranteed to find those.

Acknowledgements

We would like to thank Jason K. Johnson for suggesting us to try a case study of models with K–regular adjacency matrix and equal interaction weights, and sharing his ideas about the the walk–sum analysis of these models. The authors would like to acknowledge the support from the Vici grant 639.023.604 (second author) and Marie Curie EST program MEST-CT-2004-514510 (first author).

References

T. Heskes. Stable fixed points of loopy belief propagation are minima of the Bethe free energy. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 359–366, Cambridge, MA, 2003. The MIT Press.

T. Heskes, M. Opper, W. Wiegerinck, O. Winther, and O. Zoeter. Approximate inference techniques with expectation constraints. *Journal of Statistical Mechanics: Theory and Experiment*, 2005:P11015, 2005. URL `http://www.iop.org/EJ/jstat/`.

R. A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 2005.

T. Jaakkola. Tutorial on variational approximation methods. In M. Opper and D. Saad, editors, *Advanced mean field methods: theory and practice*, pages 129–160, Cambridge, MA, 2000. The MIT Press.

D. Malioutov, J. Johnson, and A. Willsky. Walk-sums and belief propagation in Gaussian graphical models. *Journal of Machine Learning Research*, 7:2031–2064, October 2006.

T. Minka. Power EP. Technical report, Microsoft Research Ltd., Cambridge, UK, MSR-TR-2004-149, October 2004.

T. Minka. Divergence measures and message passing. Technical report, Microsoft Research Ltd., Cambridge, UK, MSR-TR-2005-173, December 2005.

K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 467–475, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman Publishers, San Mateo, CA, 1988.

Y. Qi, M. Szummer, and T. Minka. Bayesian conditional random fields. In Robert G. Cowell and Zoubin Ghahramani, editors, *AISTATS05*, pages 269–276. Society for Artificial Intelligence and Statistics, 2005. (Available electronically at http://www.gatsby.ucl.ac.uk/aistats/).

P. Rusmevichientong and B. Van Roy. An analysis of belief propagation on the turbo decoding graph with Gaussian densities. *IEEE Transactions on Information Theory*, 47:745–765, 2001.

M. Wainwright, T. Jaakkola, and A. Willsky. Tree-reweighted belief propagation algorithms and approximate ML estimation via pseudo-moment matching. In C. Bishop and B. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics, 2003.

Y. Weiss and W. T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001.

M. Welling and Y. W. Teh. Belief optimization for binary networks: a stable alternative to loopy belief propagation. In J. S. Breese and D. Koller, editors, *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 554–561. Morgan Kaufmann Publishers, 2001.

W. Wiegerinck and T. Heskes. Fractional belief propagation. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 438–445, Cambridge, MA, 2003. The MIT Press.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems 12*, pages 689–695, Cambridge, MA, 2000. The MIT Press.

O. Zoeter and T. Heskes. Change point problems in linear dynamical systems. *Journal of Machine Learning Research*, 6:1999–2026, 2005.

Bayesian network learning by compiling to weighted MAX-SAT

James Cussens
Department of Computer Science &
York Centre for Complex Systems Analysis
University of York
Heslington, York YO10 5DD, UK
jc@cs.york.ac.uk

Abstract

The problem of learning discrete Bayesian networks from data is encoded as a weighted MAX-SAT problem and the MaxWalkSat local search algorithm is used to address it. For each dataset, the per-variable summands of the (BDeu) marginal likelihood for different choices of parents ('family scores') are computed prior to applying MaxWalkSat. Each permissible choice of parents for each variable is encoded as a distinct propositional atom and the associated family score encoded as a 'soft' weighted single-literal clause. Two approaches to enforcing acyclicity are considered: either by encoding the ancestor relation or by attaching a total order to each graph and encoding that. The latter approach gives better results. Learning experiments have been conducted on 21 synthetic datasets sampled from 7 BNs. The largest dataset has 10,000 datapoints and 60 variables producing (for the 'ancestor' encoding) a weighted CNF input file with 19,932 atoms and 269,367 clauses. For most datasets, MaxWalkSat quickly finds BNs with higher BDeu score than the 'true' BN. The effect of adding prior information is assessed. It is further shown that Bayesian model averaging can be effected by collecting BNs generated during the search.

1 Introduction

Bayesian network learning is a hard combinatorial optimisation problem which motivates applying state-of-the-art algorithms for solving such problems. One of the most successful algorithms for solving SAT, the satisfiability problem in clausal propositional logic, is the local search WalkSAT algorithm [8]. The basic idea is to search for a satisfying assignment of a CNF formula by flipping the truth-values of atoms in that CNF. Both 'directed' flips which decrease the number of unsatisfied clauses and random flips are used. Frequent random restarts ('tries') are also used.

The SAT problem can be extended to the weighted MAX-SAT problem where weights are added to each clause and the goal is to find an assignment that maximises the sum of the weights of satisfied clauses (equivalently minimises the sum of the weights of unsatisfied clauses which can then be viewed as costs). WalkSAT can be extended to MaxWalkSAT where truth value flipping is a mixture of random flips and those which aim to reduce the cost of unsatisfied clauses.

Problems of *probabilistic inference in* Bayesian networks have already been encoded as weighted MAX-SAT problems and various algorithms, including MaxWalkSAT have been applied to them [6, 7]. Similar approaches combined with MCMC have been used for inference in Markov logic [5]. The current paper appears to be the first to apply a MAX-SAT encoding to *learning of* Bayesian networks.

The paper is structured as follows. Section 2 describes the synthetic datasets used. Section 3 describes how the necessary weights were extracted from these datasets. Section 4 is the key section where the method of encoding a BN learning problem as a weighted MAX-SAT one is given. Section 5 provides empirical evaluation of the technique for both model selection and model averaging. There then follow conclusions and pointers to future work in Section 6.

All runs of MaxWalkSAT were conducted using the C implementation (sometimes slightly adapted) available from the WalkSAT home page `http://www.cs.rochester.edu/~kautz/walksat/`. All computations were performed using a 2.40GHz Pentium running under GNU/Linux.

Name	n	max $\lvert\text{Pa}\rvert$	max r
Mildew	35	3	100
Water	32	5	4
alarm	37	4	4
asia	8	2	2
carpo	60	5	4
hailfinder	56	4	11
insurance	27	3	5

Table 1: The seven Bayesian networks used in this paper. n is the number of variables, $\max|\text{Pa}|$ is the size of the biggest parent set and $\max r$ is the maximum number of values associated with a variable in the network.

Name	$N = \ldots$ 10^2	10^3	10^4
Mildew	100	1000	10000
Water	100	990	9368
alarm	94	805	5970
asia	16	30	52
carpo	100	995	9662
hailfinder	100	1000	10000
insurance	99	982	8918

Table 2: Number of distinct joint instantiations sampled from each 'true' BN where N is the total number of joint instantiations.

2 Bayesian networks and datasets

Since the goal of this paper is to evaluate a particular approach to BN learning, all datasets are the result of sampling from a known 'true' BN. Seven BNs were used: their names and key characteristics are given in Table 1. Each BN only involves discrete variables. All BNs were obtained in Bayesian Interchange (`.bif`) format from the Bayesian Network Repository (`http://compbio.cs.huji.ac.il/Repository/`). Each was then then converted into a Python class instance and 'pickled' (i.e. saved to disk).

Datasets of sizes 100, 1,000 and 10,000 were then generated by forward sampling from each BN. Note that each dataset is complete: there are no missing values. Each dataset was stored as a single table in an SQLite database. Each such table had two columns: a string representation of each distinct joint instantiation sampled and a count of how often that joint instantiation had been sampled. For big BNs most or all of these counts will be one. Table 2 states the number of distinct joint instantiations for each dataset. Each SQLite database was wrapped inside a Python class instance which was then pickled.

3 Computing and filtering family scores

The primary goal of this paper is to evaluate MaxWalkSat as an algorithm to search for BNs of high marginal likelihood for a given dataset. As is well known, if Dirichlet priors are used for the parameters of a BN and the data are complete then the marginal likelihood for a BN structure G for data D is given by the equations in (1) and (2) [3].

$$P(G|D) = L(G) = \prod_{i=1}^{n} \text{Score}_i(G) \tag{1}$$

where

$$\text{Score}_i(G) = \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(n_{ij}+\alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(n_{ijk}+\alpha_{ijk})}{\Gamma(\alpha_{ijk})} \tag{2}$$

Equation (2) defines what will be called a *family score* since it is determined by a child variable and its parents. In (2), q_i is the number of joint instantiations of those parents that X_i has in the graph G; j is thus an index over these instantiations. r_i is the number of values X_i has and thus k is an index over these values. n_{ijk} is the count of how often X_i takes its kth value when its parents are in their jth instantiation. α_{ijk} is the corresponding Dirichlet parameter. Finally, $n_{ij} = \sum_{k=1}^{r_i} n_{ijk}$ and $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$. Since $\text{Score}_i(G)$ only depends on Pa_i, the parents it has in graph G, $\text{Score}_i(G)$ can be written as $\text{Score}_i(\text{Pa}_i)$.

Throughout this paper the values of α_{ijk} have been chosen so that $L(G)$ becomes the *BDeu metric* for scoring BNs. This is done by choosing a *prior precision* value α and setting $\alpha_{ijk} = \alpha/(r_i q_i)$. In this paper the prior precision was always set to $\alpha = 1$. The BDeu score is a special case of a BDe score; BDe scores have the property that Markov equivalent BNs are guaranteed to have the same score. Setting $\alpha_{ijk} = 1/(r_i q_i)$ allows family scores to be defined by (3).

$$\text{Score}_i(\text{Pa}_i) = \prod_{j=1}^{q_i} \frac{\Gamma(\frac{1}{q_i})}{\Gamma(n_{ij}+\frac{1}{q_i})} \prod_{k=1}^{r_i} \frac{\Gamma(n_{ijk}+\frac{1}{r_i q_i})}{\Gamma(\frac{1}{r_i q_i})} \tag{3}$$

Now, let $\mathbf{X}$ be any subset of the variables and define, for fixed data, the function H, as in (4).

$$H(\mathbf{X}) = \prod_{\ell=1}^{q_{\mathbf{X}}} \frac{\Gamma(n_\ell + \frac{1}{q_{\mathbf{X}}})}{\Gamma(\frac{1}{q_{\mathbf{X}}})} \tag{4}$$

where $q_{\mathbf{X}}$ is the number of joint instantiations of variables $\mathbf{X}$, and n_ℓ is a count of how often the ℓth of these

Name	|scores|	Time taken for $N = \ldots$ 10^2	10^3	10^4
Mildew	230,300	1,678	1,942	4,502
Water	159,744	22	123	1,093
alarm	288,859	38	208	1,501
asia	512	0	0	0
carpo	2,056,800	544	1,811	13,892
hailfinder	1,555,456	852	2,974	22,666
insurance	79,704	29	97	749

Table 3: Computing all family scores with at most 3 parents. Times are in seconds rounded (asia did take a small positive amount of time!)

occurred in the data. Let Pa_i be the parents of X_i and set $\mathrm{Fa}_i = \{X_i\} \cup \mathrm{Pa}_i$ (the *family* for X_i), then it is easy to see that (3) can be rewritten as (5)

$$\mathrm{Score}_i(\mathrm{Pa}_i) = \frac{H(\mathrm{Fa}_i)}{H(\mathrm{Pa}_i)} \tag{5}$$

so that

$$\log \mathrm{Score}_i(\mathrm{Pa}_i) = \log H(\mathrm{Fa}_i) - \log H(\mathrm{Pa}_i) \tag{6}$$

In the approach followed in this paper the first stage of BN learning is to compute log family scores $\log \mathrm{Score}_i(G)$ for all families up to some limit on the number of parents. Here this limit was set to 3 parents. Note that 4 of the 7 true BNs have variables with more than 3 parents so that this restriction renders these 4 unlearnable. Equation (6) permits a small efficiency increase: $\log H(\mathrm{Fa}_i(\mathbf{X}))$ is computed for all subsets of variables from size 0 to size 4. Family scores can then be computed using (6). (All scores will be log scores from now on.)

The number of family scores computed and the time taken is listed for each dataset in Table 3. These computations were performed using Python equipped with the Psyco (`http://psyco.sourceforge.net/`) JIT compiler. Re-implementing in, say, C would no doubt be faster, but optimising this part of the process is not the focus of this paper.

Once family scores are computed the next step is to encode them as weighted clauses, but, with the exception of asia, there are too many scores for all to be used. However, since the goal is to find high likelihood BNs the vast majority of these family scores can be thrown away. Let Pa_i and Pa'_i be two potential parent sets for variable X_i. If (1) $\mathrm{Pa}_i \subset \mathrm{Pa}'_i$ and (2) Pa_i has a higher score than Pa'_i then Pa'_i can be ruled out as a potential parent set for X_i in a maximal likelihood graph. Any graph which has Pa'_i as the parents for X_i can have its score increased by replacing Pa'_i by Pa_i; the key point is that such a replacement cannot introduce a directed cycle since one or more arrows are being removed. Filtering family scores in this way causes a significant reduction in the potential parent sets for any variable as shown by Table 4.

Dat	$N = \ldots$ 10^2 max	10^2 n	10^3 max	10^3 n	10^4 max	10^4 n
Mi	977	3,515	17	163	47	465
Wa	44	482	44	573	49	961
al	75	907	184	1,928	473	6,473
as	10	41	24	107	31	161
ca	350	5,068	352	3,827	2,089	16,391
ha	22	244	77	761	435	3,768
in	28	279	95	774	496	3,652

Table 4: Results of filtering potential parents for variables. 'True' BN names have been abbreviated. n is the total number of candidate parent sets for all variables. max is the number of candidate parents for that variable which has the most candidate parents.

4 Encoding BN learning using hard and soft clauses

The goal of finding a maximum (marginal) likelihood BN with complete data is equivalent to choosing the n best parent sets (one for each variable) subject to the condition that no directed cycle is formed: a problem of combinatorial optimisation. Here this problem is encoded with weighted clauses.

A weighted CNF problem requires specifying (i) propositional atoms (ii) clauses and (iii) weights for clauses. To represent graph structure two options have been considered. Using an adjacency matrix approach for each of the $n(n-1)$ distinct ordered pairs of vertices there is an atom asserting the existence of an arc between the two vertices. Alternatively one can create one atom for each possible family (child plus parents). This latter approach has performed better in the experiments done so far, presumably because flipping the truth values of such atoms permits bigger search moves. With this approach when encoding the problem of learning from the carpo-generated dataset with 10,000 datapoints there are 16,391 such atoms (see Table 4). If such an atom is set to TRUE, this represents that the specified variable has the specified parents. For each variable there is a single clause stating that it must have a (possibly empty) parent set:

$$\bigvee_{\text{candidate parent set } t} X_i \text{ has parent set } t \tag{7}$$

The longest such clause contains 2,089 atoms.

To rule out choices of parent sets producing a cycle two options have been considered. In the first 'Ancestor' encoding, there is an atom $\mathrm{an}(X_i, X_j)$ stating that X_i is an ancestor of X_j in the graph for each of the $n(n-1)$ distinct ordered pairs of vertices. For each of the $n(n-1)(n-2)$ distinct ordered triples (X_i, X_j, X_k) there is a transitivity clause:

$$\mathrm{an}(X_i, X_j) \wedge \mathrm{an}(X_j, X_k) \rightarrow \mathrm{an}(X_i, X_k) \qquad (8)$$

To express acyclicity there are $n(n-1)/2$ clauses of the form:

$$\neg\mathrm{an}(X_i, X_j) \vee \neg\mathrm{an}(X_j, X_i) \qquad (9)$$

An alternative 'Total order' approach is to encode a total order of vertices. In a total order exactly one of $X_i < X_j$ and $X_j < X_i$ is true for each distinct X_i and X_j. So for each of the $n(n-1)/2$ distinct unordered pairs $\{X_i, X_j\}$ there is an atom $\mathrm{ord}(X_i, X_j)$ asserting that X_i and X_j are lexicographically ordered in the total order. To ensure that an assignment of truth values to these atoms defines a total order it is enough to rule out 3-cycles such as $X_i < X_j$, $X_j < X_k$, $X_k < X_i$. This is because if a fully connected digraph with no 2-cycles (a *tournament*) has a cycle then it has a 3-cycle. There is a simple inductive proof of this result which is omitted here. There are $n(n-1)(n-2)$ hard clauses ruling out 3-cycles of the form:

$$\neg\mathrm{ord}(X_i, X_j) \vee \neg\mathrm{ord}(X_j, X_k) \vee \mathrm{ord}(X_i, X_k)$$
$$\mathrm{ord}(X_i, X_j) \vee \mathrm{ord}(X_j, X_k) \vee \neg\mathrm{ord}(X_i, X_k)$$

Using either the 'Ancestor' or 'Total Order' encoding there are hard clauses declaring that a (non-empty) choice of parents for a variable determines the truth value of certain atoms. For example, using the 'Ancestor' approach:

$$X_j \text{ has parent set } \{X_i, X_k\} \rightarrow \mathrm{an}(X_i, X_j)$$
$$X_j \text{ has parent set } \{X_i, X_k\} \rightarrow \mathrm{an}(X_k, X_j)$$

or with the 'Total Order' approach:

$$X_j \text{ has parent set } \{X_i, X_k\} \rightarrow \mathrm{ord}(X_i, X_j)$$
$$X_j \text{ has parent set } \{X_i, X_k\} \rightarrow \neg\mathrm{ord}(X_j, X_k)$$

Note that since the log BDeu score is a log-likelihood it is a negative number, as are all the family scores. The goal is to find an admissible choice of families with small negative numbers. Another way of looking at this is to say that choosing any particular set of parents for a variable incurs a cost which is -1 times the relevant family score. Each such cost can then be encoded by a weighted clause of the following type:

$$-\log \mathrm{Score}_i(\mathrm{Pa}_i) : \neg(X_i \text{ has parent set } \mathrm{Pa}_i) \qquad (10)$$

The $-\log \mathrm{Score}_i(\mathrm{Pa}_i)$ costs were rounded to integers.

A final option is not to rule out cycles directly, but to create a cycle_atom asserting the existence of a cycle. Cycles can then be ruled out entirely with the hard clause ¬cycle_atom or merely discouraged with a soft version. If the latter option is taken then cyclic digraphs have to be filtered out after the search is over.

The numbers of atoms, clauses and literals for an encoding of each learning problem using a cycle_atom and the 'Ancestor' encoding is given in Table 5. Without a cycle_atom there is one fewer atom and clause and also a reduction in the number of literals. With the 'Total Order' approach there is a small reduction in atoms and the number of clauses and lits is roughly halved. (A literal is an atom or its negation, the number of literals in Table 5 is the sum of all literals in all clauses.)

To help motivate these choices of encoding it is worth peeking ahead to examine a run of MaxWalkSat with

Data	atoms	clauses	lits
Mi_2	4,706	54,367	149,123
Mi_3	1,354	40,807	122,003
Mi_4	1,656	41,579	123,547
Wa_2	1,475	32,053	94,793
Wa_3	1,566	32,194	95,075
Wa_4	1,954	33,352	97,391
al_2	2,240	50,739	149,355
al_3	3,261	53,945	155,767
al_4	7,806	70,610	189,097
as_2	98	488	1,351
as_3	164	693	1,761
as_4	218	873	2,121
ca_2	8,609	226,406	661,551
ca_3	7,368	221,365	651,469
ca_4	19,932	269,367	747,473
ha_2	3,325	170,009	509,305
ha_3	3,842	171,400	512,087
ha_4	6,849	181,545	532,377
in_2	982	18,926	56,049
in_3	1,477	20,346	58,889
in_4	4,355	30,344	78,885

Table 5: Data on the weighted CNF provided as input to MaxWalkSat using the 'Ancestor' encoding and a cycle_atom. Datasets have been abbreviated, so that e.g. in_4 is the data set of 10^4 datapoints sampled from **in**surance.

the 'Ancestor' encoding as shown in Fig 1. This is a run using the encoded form of the hailfinder-generated set of 10,000 datapoints. This is a run *without* using the cycle_atom so the numbers of atoms, clauses and literals are reduced a little from those given in Table 5. The goal was to find a BN with BDeu score at least as high as that of the 'true' BN's score of -503,040 (i.e. cost of 503,400). This was achieved in just under 13 million flips taking 75 seconds.

A key point is that the number of unsatisfied clauses in the lowest cost assignment in each try is 56, the number of variables in hailfinder. These unsatisfied clauses are 56 weighted single-literal clauses of the form (10) corresponding to a choice of parents for each variable which does not produce a cycle. In all runs on all datasets, the unsatisfied clauses in low cost assignments are *always* of this form. Choices of parents that do cause a cycle will break at least one hard clause thus incurring a sufficiently large cost that MaxWalkSat will never return them as a lowest cost assignment. Note that although only one choice of parents is possible for any variable there is no need to encode this (hard) constraint. Choosing e.g. two parent sets will always incur a higher cost then either or the two alone so the search avoids such assignments. MaxWalkSat 'wants' to avoid choosing any parents but is compelled to do so by hard constraints of the form (7).

	tries=10		tries=100	
Data	t	flip/sec	t	flip/sec
Mi_2	7	140,656	68	146,222
Mi_3	3	328,608	29	342,019
Mi_4	5	193,561	50	196,502
Wa_2	3	255,118	39	255,749
Wa_3	4	215,222	45	218,069
Wa_4	6	165,027	59	168,977
al_2	7	126,217	79	125,894
al_3	8	113,729	88	112,959
al_4	10	92,570	106	94,280
as_2	0	1,923,202	5	1,977,716
as_3	0	1,676,086	6	1,533,842
as_4	0	1,546,493	6	1,548,088
ca_2	17	55,963	175	56,821
ca_3	16	60,548	168	59,500
ca_4	23	43,230	236	42,287
ha_2	8	124,283	79	126,143
ha_3	14	71,195	141	70,438
ha_4	15	65,938	154	64,905
in_2	2	476,978	20	498,619
in_3	2	371,311	27	359,390
in_4	4	217,090	45	217,800

Table 7: Timings corresponding to 'Ancestor' encoding runs with a 'cycle_atom' and noise at 50%. t is the total time taken in seconds (rounded). Flip/sec is the frequency of literal flipping performed by MaxWalkSat.

5 Results

5.1 Learning a single BN

In this section, results using MaxWalkSat to search for a high BDeu-scoring BN are given. The scores from these searches are given in Table 6 and some timings are given in Table 7. Note that the times reported in Table 7 are for one of the slowest approaches: 'Ancestor' encoding, 50% noise (random flips). An example of such a 'slow' run was shown in Fig 1 where only 171,206 flips/sec were achieved. In contrast, using the 'Total Order' encoding with 10% noise on the same data a rate of 1,092,243 flips/secs is achievable.

One problem with evaluating a search for a high scoring BN is that the optimal BN and its score is unknown (although presumably it could be computed for the small asia learning problem). However, by choosing to evaluate on synthetic data there is at least the 'true' BN which can be scored. It is also possible to construct a high-scoring BN which meets the restriction of having at most 3 parents. Starting with the true BN replace each true parent set with the highest-scoring pre-scored parent set which is a subset of the true parent set. With the exception of carpo this produces a BN with a higher score than the true BN. The score of the BN thus constructed is given in the 'Target' column of Table 6.

Table 6 contains only the most interesting of many experimental runs conducted. The results show that for small datasets (*_2, corresponding to $10^2 = 100$), MaxWalkSat easily finds BNs exceeding both the True and 'Target' BN score. This reflects the relatively smooth nature of the likelihood surface. For bigger datasets, were the landscape is much more jagged, the picture is more mixed. For 10^3 size datasets the True BN score is beaten by the 'Long' run in all cases, except for alarm. For 10^4, the 'Long' run fails to beat carpo and insurance also. Comparing 'Ancestor' to 'Total Order' results show that the latter is the more successful encoding. Experiments unreported here have indicated that the 'Total Order' encoding works better with low noise (random flip) values and a prevention of random breaking of hard clauses. Combining these with an increased search on each try leads to the superior results in the 'Long' column of Table 6. Note that even the longest 'Long' run, ca_4, only took 32 minutes (with 510,397 flips/sec); most where much faster.

Note that in some cases MaxWalkSat returns a BN

```
newmaxwalksat version 20 (Huge)
seed = 99955222
cutoff = 10000000
tries = 100
numsol = 1
targetcost = 503040
heuristic = best, noise 50 / 100, init initfile
allocating memory...
clauses contain explicit costs
numatom = 6848, numclause = 181544, numliterals = 529296
wff read in
                                        average             average       mean
    lowest     worst    number             when               over      flips
      cost    clause    #unsat    #flips    model   success     all      until
  this try  this try  this try  this try    found      rate    tries     assign
    506076     16968        56  10000000        *         0        *          *
    501973     23318        56   2913803  2913803        50 12913803 12913803.0

total elapsed seconds = 75.428415
average flips per second = 171206
number of solutions found = 1
mean flips until assign = 12913803.000000
mean seconds until assign = 75.428415
mean restarts until assign = 2.000000
ASSIGNMENT ACHIEVING TARGET 503040 FOUND
```

Figure 1: A successful search (using the 'Ancestor' encoding) for a BN exceeding the BDeu score for the hailfinder BN using data of 10,000 data points sampled from that BN.

Data	True	Target	Ancestor	Total order	Long	> True
Mi_2	-7,786	-6,611	-5,711	-5,708	-5,705	Y
Mi_3	-63,837	-52,866	-47,229	-47,194	-47,120	Y
Mi_4	-470,215	-459,874	-409,641	-410,159	-408,282	Y
Wa_2	-1,801	-1,523	-1,488	-1,486	-1,484	Y
Wa_3	-13,843	-13,304	-13,293	-13,284	-13,247	Y
Wa_4	-129,655	-128,745	-129,274	-128,916	-128,812	Y
al_2	-1,410	-1,383	-1,368	-1,368	-1,336	Y
al_3	-11,305	-11,255	-11,599	-11,501	-11,339	N
al_4	-105,303	-105,226	-107,205	-106,503	-105,907	N
as_2	-247	-245	-241	-241	-241	Y
as_3	-2,318	-2,318	-2,312	-2,312	-2,312	Y
as_4	-22,466	-22,466	-22,462	-22,462	-22,462	Y
ca_2	-1,969	-1,952	-1,849	-1,852	-1,824	Y
ca_3	-17,739	-17,821	-17,938	-17,891	-17,731	Y
ca_4	-173,682	-175,349	-175,832	-176,456	-174,605	N
ha_2	-6,856	-6,058	-5,998	-5,998	-5,991	Y
ha_3	-55,366	-52,747	-53,059	-52,881	-52,517	Y
ha_4	-503,040	-498,163	-506,219	-503,420	-498,739	Y
in_2	-1,951	-1,715	-1,690	-1,689	-1,674	Y
in_3	-14,411	-13,919	-14,105	-14,121	-13,934	Y
in_4	-133,489	-132,968	-134,378	-134,730	-133,841	N

Table 6: Scores for true and target BNs (first 2 columns) followed by scores achieved using the Ancestor and Total Order encodings with 50% noise and 100,000 flips per try. 'Long' indicates using the Total Order encoding and runs using $10,000,000$ flips per try, noise at 10% and disallowing the breaking of hard clauses by random flips. In all 3 cases 100 tries (restarts) were done. The final column indicates whether the 'Long' run produced a BN with a score exceeding that of the true BN.

which is astronomically more probable (assuming a uniform prior) than the 'true' one. For example, with Mi_2 and the Ancestor encoding, the returned BN is $e^{(-470,215+409,641)} = e^{60,574}$ more probable than the true one!

A rough comparison with Castelo and Kočka's RCARNR and RCARR algorithms [1] is possible for the al_2 and al_4 datasets. As here, datasets of size 1,000 and 10,000 were sampled from the Alarm BN and best scores of around -11,115 and -108,437 respectively were reported. This beats (resp. does not beat) our best score for Alarm, but since the actual sampled datasets are different not too much should be read into this comparison.

Some initial experiments adding prior knowledge to the 'Ancestor' encoding proved interesting. Adding in information specifying *all* known pairs of independent variables (by banning the relevant ancestor relations) had very little effect on the scores achieved. On the other hand, setting just a small number of known ancestor relations improved scores considerably. This is presumably due to the search having a bias towards sparser (less constrained) BNs.

5.2 Bayesian model averaging by search

Markov chain Monte Carlo (MCMC) approaches are popular for Bayesian model averaging (BMA) for Bayesian networks. The idea is to construct a Markov chain whose stationary distribution is the true posterior over BNs. By running the chain for a 'sufficient' number of iterations (and throwing away an initial burn-in) the hope is that the sample thus produced supplies a reasonable approximation to the true distribution.

Success depends on a realisation of the chain wandering into areas of high probability and visiting different BNs with a relative frequency close to the BNs' posterior probabilities. A search-based approach offers a much more direct method of BMA. The key idea is to explicitly search for areas of high probability. If, as is the case here, the probability of any visited BN can be computed at least up to a normalising constant then this value is recorded. An estimate of the posterior distribution is then found by simply assigning zero probability to any BN not encountered in the search and performing the obvious renormalisation on BN scores. An early paper proposing search for BMA is [4].

Here a crude search-based approach using MaxWalkSat and the 'Ancestor' encoding has been tried out. MaxWalkSat is run as normal and for *every* assignment where no hard clause is broken a record of the n parent sets given by that assignment together with the cost of the assignment is sent to standard output. Piping this through UNIX's `sort --unique` is enough to produce a list of distinct (encoded) BNs sorted by score.

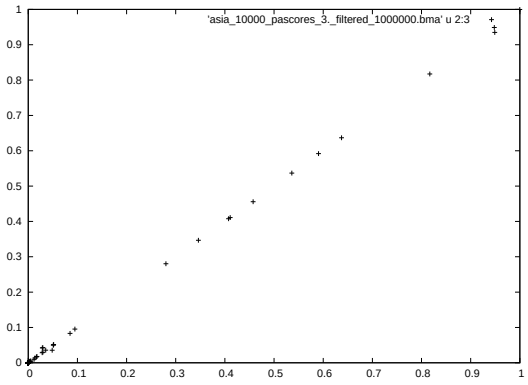


Figure 2: Comparing estimated posterior probabilities from two different BMA runs using MaxWalkSAT using asia 10,000 data

The distribution thus created can be used to estimate true posterior quantities. In experiments conducted so far only the posterior probabilities of parent sets have been estimated. By comparing estimates produced from two runs, some measure of success can be produced. Here each run used the 'Ancestor' encoding and came from 120 restarts of MaxWalkSat each using 100,000 flips. Only the 1,000,000 highest scoring BNs from each run were kept.

For the datasets considered here only asia (all sizes) and Water with 100 datapoints afforded successful BMA. Plots comparing estimated posterior probabilities for parent sets for these two cases are given in Figs 2 and 3. In all other cases such plots show big differences in estimated probabilities for most cases where the estimates differ significantly from zero. Fig 4 shows a typical excerpt of such a failure using the alarm 100 data.

The basic problem here is that, apart from the trivial asia problem, the highest scoring one or two BNs found on any run are often vastly more probable than any others, so that the estimate of the posterior distribution puts almost all the probability mass on one, occasionally two, BN(s). If these few BNs vary between runs, estimates produced from them generally vary also. Fig 5 shows a typical example of the top BN in a run being much more probable than the second-ranking one. The central issue is that often the true posterior probability really is concentrated on a few BNs, so BMA either needs to reliably find those few

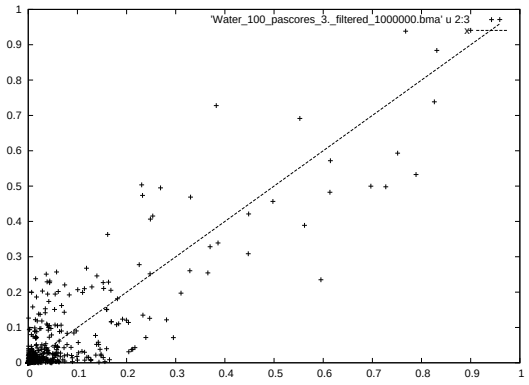


Figure 3: Comparing estimated posterior probabilities from two different BMA runs using MaxWalkSAT using Water 100 data

```
1622 9.89412087285e-35 5.71801239082e-28
1793 0.0013149823888 0.465138403992
1792 1.51462961172e-20 5.38296345824e-18
1791 5.89129801134e-14 2.10283310245e-06
1398 0.924408916903 0.00117720289971
1797 1.46223426395e-06 1.77652663003e-11
```

Figure 4: Estimates of posterior probability of parent sets from two different BMA runs for Alarm $N = 100$ dataset. The first column is the integer identifier for the parent set used by MaxWalkSat.

models or needs to move in a more regular space, such as the space of variable orderings [2].

6 Conclusions

The results given here should be seen as a preliminary exploration of the worth of weighted MAX-SAT encodings of BN learning and MaxWalkSat as a method for solving such encoded problems. In particular, future work needs to consider alternative encodings and greater incorporation of prior knowledge on structures. Exponential-family structure priors whose features can be easily encoded are an obvious choice; they can be encoded by weighted clauses, just as the family scores are. This paper has focused on BN learning as a pure BDeu score optimisation problem. Further work will consider the value of the learned BNs according to other metrics and provide a comparison of this learning approach with existing BN learning systems.

```
C 176696  3954 4048 4197 4262   ... 17333 ...
C 176714  3954 4048 4197 4262   ... 17332 ...
```

Figure 5: Top two BNs from a BMA run using the carpo $N = 10,000$ data. First number is the (rounded, negated) BDeu score. First one is 65 million times more probable (assuming a uniform prior) than second even though they differ in only one parent set (17333 vs. 17332).

How to reproduce these results Please go to: `http://www.cs.york.ac.uk/~jc/research/uai08/`

Acknowledgements

Thanks to those responsible for hosting the Bayesian Network Repository and to Henry Kautz for making MaxWalkSat available. Thanks also to 3 anonymous reviewers.

References

[1] Robert Castelo and Tomáš Kočka. On inclusion-driven learning of Bayesian networks. *Journal of Machine Learning Research*, 4:527–574, 2003.

[2] Nir Friedman and Daphne Koller. Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50:95–126, 2003.

[3] David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[4] David Madigan and Adrian E. Raftery. Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, 89:1535–1546, 1994.

[5] Hoifung Poon and Pedro Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proc. AAAI-06*, pages 458–463, 2006.

[6] Tian Sang, Paul Beame, and Henry Kautz. Solving Bayesian inference by weighted model counting. In *Proc. AAAI-05*, Pittsburgh, 2005.

[7] Tian Sang, Paul Beame, and Henry Kautz. A dynamic approach to MPE and weighted MAX-SAT. In *Proc. IJCAI-07*, Hyderabad, 2007.

[8] Bart Selman, Henry Kautz, and Bram Cohen. Local search strategies for satisfiability testing. In David S. Johnson and Michael A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*. AMS, 1993.

Identifying Optimal Sequential Decisions

A. Philip Dawid
Statistical Laboratory,
Centre for Mathematical Sciences,
Cambridge, UK
apd@statslab.cam.ac.uk

Vanessa Didelez
Department of Mathematics
University of Bristol, UK
vanessa.didelez@bristol.ac.uk

Abstract

We consider conditions that allow us to find an optimal strategy for sequential decisions from a given data situation. For the case where all interventions are unconditional (atomic), identifiability has been discussed by Pearl & Robins (1995). We argue here that an optimal strategy must be conditional, i.e. take the information available at each decision point into account. We show that the identification of an optimal sequential decision strategy is more restrictive, in the sense that conditional interventions might not always be identified when atomic interventions are. We further demonstrate that a simple graphical criterion for the identifiability of an optimal strategy can be given.

1 INTRODUCTION

Consider the case of a chronically ill patient who regularly sees their doctor in order to adjust their treatment to the individual development of their disease. For example patients who receive anticoagulation treatment are subject to regular blood testing; depending on the outcome of such a test, and possibly on earlier blood tests, the dosage of the anticoagulant might be modified. An optimal strategy in this context is a rule that stipulates, for each point in time where an intervention is carried out, a dosage to be given so as to optimise the treatment outcome, e.g. to keep the coagulation measure stable within certain bounds. It is intuitively obvious that to achieve optimality such a rule will typically have to be a function of the individual patient's history, e.g. for a patient whose blood coagulates too fast the dose has to be increased while for a patient whose blood coagulates too slowly it has to be decreased. Decision strategies where each intervention is allowed to depend on the individual history are called *conditional*, *dynamic* or *adaptive treatment* strategies. A particular feature of such strategies is that for a given patient it is not known what exact dose they will receive at future interventions as this will depend on their future coagulation test results which are subject to random variation and influenced by many factors other than treatment, such as diet and lifestyle.

Here we consider the question of what kind of data situation will allow us to construct, i.e. identify, an optimal decision strategy. This is particularly relevant when data is obtained from observational studies, but also informs us about the design of experimental studies. Essentially we need to be able to identify all conditional strategies over which we want to optimise. Identifiability of unconditional sequential strategies (atomic interventions) has been considered by Pearl & Robins (1995), where a graphical criterion is given to read identifiability off a causal diagram. Dawid & Didelez (2005) generalise their graphical criterion to the case where some or all of the interventions are allowed to depend on some or all of the observable previous information. Our main result here is that if *all* interventions are allowed to depend on *all* of the observable previous information, as would be required to find an optimal strategy, then the graphical check simplifies considerably and we only need to check what is called *simple stability* (Dawid & Didelez, 2005).

Motivated by the question of identifying conditional (sequential) interventions, the identification of *conditional interventional distributions* has been considered (e.g. Pearl, 2000, section 4.2; Tian, 2004; Shpitser & Pearl, 2006) within the context of causal diagrams where all hidden variables are represented implicitly using bidirected edges. In particular, Shpitser & Pearl (2006) give necessary and sufficient criteria for this identification problem. Our approach is slightly different as we use influence diagrams, where the interventions are made explicit by suitable decision nodes and where unobservable variables are also explicitly shown by individual nodes (Dawid, 2002).

We do not consider in this paper the actual estimation and optimisation procedure required to find an optimal strategy, which is a numerically demanding task and a topic of its own. A regret–based approach to this has been proposed by Murphy (2003), while Robins (2004) suggests structural nested models (cf. also discussion by Moodie et al. (2007)). An application of the regret–based method to the anticoagulation problem described above can be found in Rosthoj et al. (2008).

The paper is organised as follows. In section 2 we introduce the notation used throughout. Optimal strategies are addressed in section 3. The general problem of identifying possibly conditional strategies is covered in section 4, where we present a simple sufficient criterion called simple stability as well as a more general criterion and explain how both can be checked graphically using influence diagrams. The latter reduces to simple stability when we consider optimal strategies as shown in section 5. We discuss and compare our approach in sections 6 and 7.

2 GENERAL SET–UP

Let $A_1, \ldots, A_N$ be variables that can be set by some intervention (action variables), while $L_1, \ldots, L_N$ (each of which can be vector valued) are additional observations / covariates, and let $L_{N+1} \equiv Y$ be the outcome of interest. To simplify exposition we restrict ourselves to the case of all variables being discrete. We assume that $\mathbf{L}^{\leq i} = (L_1, \ldots, L_i)$ can be observed before a decision about action A_i is made. We will denote $\mathbf{L} = \mathbf{L}^{\leq N} = (L_1, \ldots, L_N)$ and similar for $\mathbf{A}$. A strategy $\mathbf{s} = (s_1, \ldots, s_N)$ consists of a set of functions that assign to any partial history $(\mathbf{a}^{<i}, \mathbf{l}^{\leq i}) = (a_1, \ldots, a_{i-1}, l_1, \ldots, l_i)$ a value $a_i = s_i(\mathbf{a}^{<i}, \mathbf{l}^{\leq i})$ in the space of A_i, and by following strategy $\mathbf{s}$ we mean that A_i are set to $s_i(\mathbf{a}^{<i}, \mathbf{l}^{\leq i})$ by some intervention for all $i = 1, \ldots, N$. Let $\mathcal{S}$ be the set of relevant strategies (this set might be restricted e.g. due to feasibility). As mentioned above, a strategy where $s_i(\mathbf{a}^{<i}, \mathbf{l}^{\leq i}) \equiv s_i$ does not actually depend on $\mathbf{a}^{<i}, \mathbf{l}^{\leq i}$ is called unconditional, as the actual values that the A_i's are set to do not depend on the observed history. Otherwise we call the strategy conditional. More generally we might also allow random (or stochastic) strategies, where $s_i(\mathbf{a}^{<i}, \mathbf{l}^{\leq i})$ specifies a distribution over the space of A_i meaning that the intervention consists of drawing a_i from this distribution and then setting $A_i = a_i$ by an intervention. Our framework allows for such randomised strategies (cf. Didelez et al. (2006) where this is used in a similar context, focusing on direct effects).

3 OPTIMAL STRATEGIES

If the distribution $p(y; \mathbf{s})$ of Y when following strategy $\mathbf{s}$ is known, we can evaluate for any function $k(\cdot)$ the expectation $E\{k(Y); \mathbf{s}\}$; typically $k(\cdot)$ would be a loss function. This calculation can be implemented recursively. Define

$$f(\mathbf{a}^{\leq j}, \mathbf{l}^{\leq i}) = E\{k(Y) | \mathbf{a}^{\leq j}, \mathbf{l}^{\leq i}; \mathbf{s}\}$$

where $j = i$ or $j = i-1$ and $i = 1, \ldots, N+1$[1]. We have that $f(\mathbf{a}^{\leq N}, \mathbf{l}^{\leq N+1}) = k(y)$ and $f(\emptyset) = E\{k(Y); \mathbf{s}\}$, where starting with the former the latter can be obtained by iteratively applying the following operations for $i = N+1, \ldots, 1$

$$f(\mathbf{a}^{<i}, \mathbf{l}^{\leq i}) = \sum_{a_i} p(a_i | \mathbf{a}^{<i}, \mathbf{l}^{\leq i}; \mathbf{s}) \times f(\mathbf{a}^{\leq i}, \mathbf{l}^{\leq i}) \quad (1)$$

(note that $p(a_i | \mathbf{a}^{<i}, \mathbf{l}^{\leq i}; \mathbf{s})$ is by definition determined by the strategy $\mathbf{s}$ and equal to the indicator function $\mathbf{I}\{a_i = s_i(\mathbf{a}^{<i}, \mathbf{l}^{\leq i})\}$ unless $\mathbf{s}$ is a randomised strategy) and

$$f(\mathbf{a}^{<i}, \mathbf{l}^{<i}) = \sum_{l_i} p(l_i | \mathbf{a}^{<i}, \mathbf{l}^{<i}; \mathbf{s}) \times f(\mathbf{a}^{<i}, \mathbf{l}^{\leq i}). \quad (2)$$

This is exactly the procedure underlying the "extensive form" analysis of sequential decision theory (see e.g. Raiffa (1968)).

The optimal strategy $\mathbf{s}_{opt}$ is given by optimising $E\{k(Y); \mathbf{s}\} = f(\emptyset)$ over the set $\mathcal{S}$; assuming that $k(\cdot)$ is chosen such that large values are better, we have

$$\mathbf{s}_{opt} = \arg\max_{\mathcal{S}} E\{k(Y); \mathbf{s}\}.$$

The dynamic programming method to find an optimal strategy essentially proceeds as follows: starting with $i = N+1$ and working down to $i = 1$ find a_i as function of $(\mathbf{a}^{<i}, \mathbf{l}^{\leq i})$ that maximises $f(\mathbf{a}^{\leq i}, \mathbf{l}^{\leq i})$. Hence, the optimal strategy will typically be conditional on past observations.

4 IDENTIFIABILITY

In practice we do not know the conditional distributions $p(l_i | \mathbf{a}^{<i}, \mathbf{l}^{<i}; \mathbf{s})$, $i = 1, \ldots, N+1$ for all $\mathbf{s} \in \mathcal{S}$ required to evaluate (2). Identifiability addresses the question whether data that have been gathered under an observational regime (which might be a sequentially randomised trial, or on observational study in the traditional meaning) can, in principle, inform us about these conditional distributions. We first address the

[1] If the subscript or superscript of a set is not defined then the set is defined to be $\emptyset$

question of identifiability of a single strategy $\mathbf{s}$. It then follows that the optimal strategy can be identified if all strategies in $\mathcal{S}$ are identified so that the above optimisation can be carried out.

In order to formalise the question of identification we introduce an indicator variable σ for the regime, where $\sigma = o$ denotes the observational regime, under which the data is collected, and $\sigma = \mathbf{s}$ denotes the regime under which strategy $\mathbf{s}$ is followed. In a sequentially randomised study $p(a_i|\mathbf{a}^{<i}, \mathbf{l}^{\leq i}; o)$ would typically be known, whereas in a traditional observational study it would be unknown but might be estimable from data.

Identifiability: A strategy $\mathbf{s}$ is identified if we can obtain $E\{k(Y); \mathbf{s}\}$ uniquely from the joint distribution of $(\mathbf{A}, \mathbf{L}, Y)$ under $\sigma = o$.

Conditions for identifiability require a positivity condition, such that actions that arise from the strategy $a_i = s_i(\mathbf{a}^{<i}, \mathbf{l}^{\leq i})$ actually have a positive probability to occur under $\sigma = o$ if the history $(\mathbf{a}^{<i}, \mathbf{l}^{\leq i})$ has a positive probability to occur under $\sigma = \mathbf{s}$ (for a more formal definition of positivity see Dawid & Didelez (2005)); here, we will take positivity for granted.

4.1 SIMPLE STABILITY

It can easily be seen from (2) that a sufficient condition for identifiability of $\mathbf{s}$ is that the following conditional distributions are the same under both regimes

$$p(l_i|\mathbf{a}^{<i}, \mathbf{l}^{<i}; \mathbf{s}) = p(l_i|\mathbf{a}^{<i}, \mathbf{l}^{<i}; o),$$

$i = 1, \ldots, N+1$, whenever the conditioning event has positive probability under both regimes (Dawid & Didelez, 2005). This is called *simple stability* and also symbolised by

$$L_i \perp\!\!\!\perp \sigma | (\mathbf{A}^{<i}, \mathbf{L}^{<i}), \qquad i = 1, \ldots, N+1 \tag{3}$$

where σ takes values in $\{o, \mathbf{s}\}$. Simple stability is closely related to the *no unmeasured confounders* assumption (Robins, 1997). Note that the above notation for conditional independence (Dawid, 1979) has been generalised for problems involving decision variables by Dawid (2002), and can be checked graphically[2] on appropriate influence diagrams as shown in Figure 1, for example.

4.2 G–RECURSION

With simple stability, the target $E\{k(Y); \mathbf{s}) = f(\emptyset\}$ can now be obtained by iterating (1) and (2) starting with $i = N+1$, where in (1) $p(a_i|\mathbf{a}^{<i}, \mathbf{l}^{\leq i}; \mathbf{s})$ is known by definition of the strategy $\mathbf{s}$ and (2) can be modified to

$$f(\mathbf{a}^{<i}, \mathbf{l}^{<i}) = \sum_{l_i} p(l_i|\mathbf{a}^{<i}, \mathbf{l}^{<i}; o) \times f(\mathbf{a}^{<i}, \mathbf{l}^{\leq i}).$$

due to simple stability. The name G–recursion has been coined by Robins (1986).

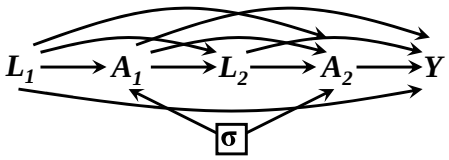


Figure 1: Example for simple stability.

4.3 EXTENDED STABILITY

It will typically be difficult to believe in simple stability (3) without additional considerations. We might want to proceed by constructing a more complex but more acceptable model, typically including additional not necessarily observable variables, and investigate whether or not we can deduce the desired simple stability property. We denote the set of additional variables by $\mathbf{U}$. *Extended stability* then holds if this set can be partitioned into $(U_1, \ldots, U_N)$ (each U_i can be vector valued or the empty set) such that $\mathbf{U}^{<i}$ are not affected by an intervention in A_i, $i = 1, \ldots, N$, and

$$(L_i, U_i) \perp\!\!\!\perp \sigma | (\mathbf{A}^{<i}, \mathbf{L}^{<i}, \mathbf{U}^{<i}), \quad i = 1, \ldots, N+1, \tag{4}$$

Clearly (4) implies that strategy $\mathbf{s}$ could be identified if $\mathbf{U}$ was observed as it is then just the same as simple stability w.r.t. $(\mathbf{A}, \mathbf{L}, \mathbf{U}, Y)$.

In many problems an extended stability assumption might be regarded as more reasonable and defensible than simple stability, so long as appropriate unobserved variables $\mathbf{U}$ are taken into account. For example, this might be the case if we believed that under $\sigma = o$ the actions A_i were taken by a decision-maker who had access to variables in the set $\mathbf{U}$ as well as $\mathbf{L}$.

Extended stability does not in general allow G–recursion if $\mathbf{U}$ is unobserved. However, it may do so if we can assume an (in)dependence structure on $(\mathbf{A}, \mathbf{L}, \mathbf{U}, Y)$ and σ that allows us to deduce that simple stability (3) does hold. Dawid & Didelez (2005) give some further conditions for this and show how these can be verified graphically. As an example consider the graphs in Figure 2, where extended stability holds (here $L_1 = \emptyset$). In graph (a) simple stability is violated as $L_2 \not\perp\!\!\!\perp \sigma | A_1$. In graph (b) the only change is

[2] using the *moralisation* criterion (cf. Cowell et al., 1999) or, equivalently, *d–separation* (Verma & Pearl, 1990)

that $U_1 = L_1$ can now be observed and then simple stability holds as $L_1 \perp\!\!\!\perp \sigma$, $L_2 \perp\!\!\!\perp \sigma | A_1, L_1$ and $Y \perp\!\!\!\perp \sigma | \mathbf{A}, \mathbf{L}$.

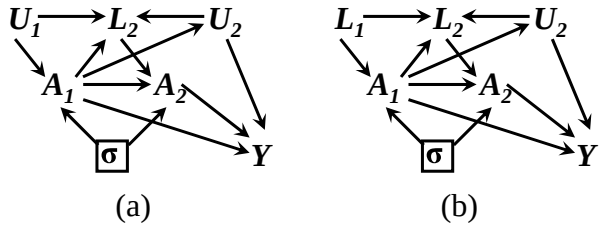


Figure 2: Examples for extended stability; in (a) simple stability is violated while in (b) it holds.

4.4 GENERAL CONDITIONS

A strategy $\mathbf{s}$ can be identified under weaker conditions than simple stability. This has been demonstrated by Pearl & Robins (1995) for the case of unconditional strategies. Next we extend the result to conditional (possibly stochastic) strategies.

We assume extended stability with respect to $(\mathbf{A}, \mathbf{L}, \mathbf{U}, Y)$ and definejoint distributions $p_i(\cdot)$ as

$$p_i(\mathbf{A}, \mathbf{L}, \mathbf{U}, Y) = p(\mathbf{A}^{\leq i}, \mathbf{L}^{\leq i}, \mathbf{U}^{\leq i}; o) \times p(\mathbf{A}^{>i}, \mathbf{L}^{>i}, \mathbf{U}^{>i}, Y | \mathbf{A}^{\leq i}, \mathbf{L}^{\leq i}, \mathbf{U}^{\leq i}; \mathbf{s})$$

Hence $p_0(\cdot) = p(\cdot; \mathbf{s})$ is the joint distribution under the strategy $\mathbf{s}$, while $p_N(\cdot) = p(\cdot; o)$ is the joint distribution under the observational regime, where we exploit extended stability to obtain $p(Y | \mathbf{A}, \mathbf{L}, \mathbf{U}; \mathbf{s}) = p(Y | \mathbf{A}, \mathbf{L}, \mathbf{U}; o)$.

Theorem 1. Under extended stability, the strategy $\mathbf{s}$ is identified by G–recursion if

$$p_{i-1}(y | \mathbf{a}^{\leq i}, \mathbf{l}^{\leq i}) = p_i(y | \mathbf{a}^{\leq i}, \mathbf{l}^{\leq i}), \qquad i = 1, \ldots, N. \quad (5)$$

Proof: see appendix, and Dawid & Didelez (2005).

Property (5) can be paraphrased as the distribution of Y given $\mathbf{a}^{\leq i}, \mathbf{l}^{\leq i}$ having to be the same regardless of whether a_i has arisen out of the strategy $\mathbf{s}$ or from observation o, when we know that future actions will follow the strategy $\mathbf{s}$.

A graphical check for (5) is more involved than for simple stability as it has to reflect the particular construction of the distributions p_i. This will be addressed in the next section.

4.5 GRAPHICAL CHECKS

If we express our subject matter background knowledge about the conditional independence structure among $(\mathbf{A}, \mathbf{L}, \mathbf{U}, Y)$ and the way we can intervene in $\mathbf{A}$ graphically we can check the above conditions for identifiability by simple graphical checks. Two approaches are possible. Firstly, we can augment the graph with the intervention indicator σ as advocated in Pearl (1993), Lauritzen (2001), Dawid (2002); this augmented graph (influence diagram) will be denoted by D. Simple stability (3) wrt. observables $(\mathbf{A}, \mathbf{L}, Y)$ can, for example, be checked on such influence diagrams as in Figures 1 and 2. Secondly, and as is common in much of the mainstream causal literature, we can take the interventions in $\mathbf{A}$ as implicit and formulate graphical conditions involving $(\mathbf{A}, \mathbf{L}, \mathbf{U}, Y)$ only, omitting σ. We denote the graph which implicitly assumes extended stability with respect to sequential interventions in $\mathbf{A}$ by D' (this is also called a causal graph with respect to $\mathbf{A}$ (Pearl, 2000). The graphs D and D' only differ in that the former has the additional decision node σ with arrows into $\mathbf{A}$. It is easy to see that simple stability can therefore be checked on D' by assessing whether $\mathbf{L}^{<i}$ satisfies the back–door criterion relative to $(\mathbf{A}^{<i}, L_i)$ (Pearl, 1995). This implies that the causal effects of each A_i on later covariates L_j, $j > i$, are identified.

For the graphical check of (5) we first define the different parent sets for the actions A_i under different regimes. Let $\mathrm{pa}_o(A_i)$ be the parent nodes (excluding σ) of A_i in D when $\sigma = o$ and let $\mathrm{pa}_{\mathbf{s}}(A_i)$ be the parent nodes (excluding σ) of A_i in D when $\sigma = \mathbf{s}$, i.e. if $s_i(\mathbf{a}^{<i}, \mathbf{l}^{\leq i})$ is constant in some of its arguments then these are not in $\mathrm{pa}_{\mathbf{s}}(A_i)$. The two parent sets are not the same, as under $\sigma = o$ A_i may depend on some variables in $\mathbf{U}^{\leq i}$, while under a strategy $\mathbf{s}$ we can obviously only take observable variables into account when choosing an action.

Now, we construct augmented graphs D_i such that the only arrow out of the node σ is into A_i, and such that the graph parents of the action variable A_j are given by $\mathrm{pa}_o(A_j)$ for $j < i$ and $\mathrm{pa}_{\mathbf{s}}(A_j)$ for $j > i$ while the parent set for A_i is the union of the parents under both regimes and σ. Such a graph represents the factorisation of the distribution p_i constructed in section 4.4 if A_i arises under $\sigma = o$ and of p_{i-1} if A_i is generated according to $\sigma = \mathbf{s}$. Let $[\cdot \perp\!\!\!\perp \cdot | \cdot]_{D_i}$ denote graph separation in D_i then we have that (5) holds if

$$[Y \perp\!\!\!\perp \sigma | \mathbf{A}^{\leq i}, \mathbf{L}^{\leq i}]_{D_i} \quad (6)$$

This procedure is illustrated in Figure 3 for the example of graph (a) from Figure 2 assuming an unconditional intervention in A_2, i.e. $\mathrm{pa}_{\mathbf{s}}(A_2) = \emptyset$. Hence there are no arrows into A_2 in D_1. We can easily see that $[Y \perp\!\!\!\perp \sigma | A_1]_{D_1}$ and $[Y \perp\!\!\!\perp \sigma | A_1, A_2, L_2]_{D_2}$ showing that a strategy where A_2 is chosen without taking

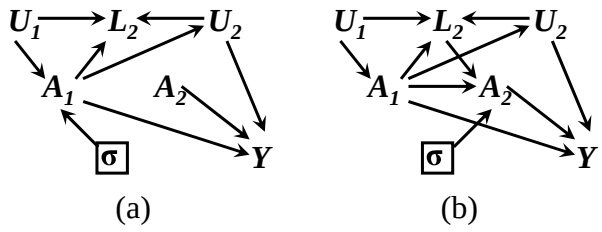


Figure 3: Same example as in Figure 2(a); here (a) shows D_1 and (b) shows D_2 with uncond. intervention in A_2.

past covariates into account *is* identifiable even though simple stability as investigated in Figure 2 is violated.

In contrast, in the same example if we assume that the intervention s_2 in A_2 *does* depend on previous covariates, i.e. $\text{pa}_{\mathbf{s}(A_2)} = (A_1, L_2)$ then we have to modify D_1 as shown in Figure 4. Now we find that $[Y \perp\!\!\!\perp \sigma | A_1]_{D_1}$ is violated and we cannot guarantee that such a conditional strategy is identifiable.

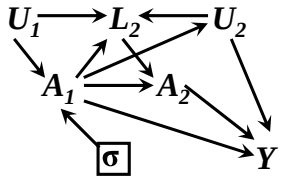


Figure 4: Same example as in Figure 3(a), D_1 with conditional intervention in A_2.

Pearl & Robins (1995) show that based on a causal diagram D' (that does not include a node σ) and for unconditional sequential interventions sufficient graphical conditions are given by

$$[Y \perp\!\!\!\perp A_i | \mathbf{A}^{<i}, \mathbf{L}^{\leq i}]_{D'_i}, \tag{7}$$

where D'_i is the graph D' with all edges out of A_i and all edges into $\mathbf{A}^{>i}$ removed. Comparing this with our procedure we can see that the idea is the same: deleting the edges out of A_i corresponds to retaining only the back–door paths from A_i as it is only these that are relevant when checking (6) due to σ having only an arrow into A_i. Further, deleting every edge into $A_{i+1}, \ldots, A_N$ corresponds to changing the parent sets of these variables to only include the parents under $\mathbf{s}$. Note that if the interventions are unconditional then A_j has no parents among $\mathbf{A}^{<j}, \mathbf{L}^{\leq j}, \mathbf{U}^{\leq j}$ under $\sigma = \mathbf{s}$.

Hence, an immediate extension of Pearl & Robins' approach, that has also been suggested by Robins (1997), to the case of conditional interventions is given by modifying the meaning of D'_i in (7) so that only edges into A_j, $j = i+1, \ldots, N$, are deleted that the intervention s_j does not depend on. This will always be the case for all edges from $\mathbf{U}^{\leq j}$ into A_j because the intervention can obviously not be a function of unobserved quantities. This is illustrated in Figure 5 for the same example as above with a conditional intervention at A_2. We can see that $[Y \perp\!\!\!\perp A_1]_{D'_1}$ is violated while $[Y \perp\!\!\!\perp A_2 | A_1, L_2]_{D'_2}$ holds.

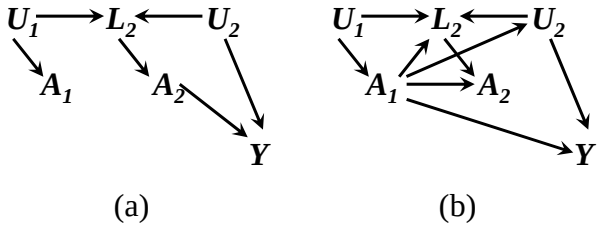


Figure 5: Pearl & Robins' check when s_2 is conditional, (a) D'_1 and (b) D'_2

5 IDENTIFIABILITY OF OPTIMAL STRATEGIES

As argued in section 3, the optimal strategy will typically be a conditional strategy, i.e. a strategy that must allow action $a_i = s_i(\mathbf{a}^{<i}, \mathbf{l}^{\leq i})$ to depend on all of $(\mathbf{a}^{<i}, \mathbf{l}^{\leq i})$ (notice that dependence on $\mathbf{a}^{<i}$ is only relevant for random strategies as otherwise a_j is a function of $\mathbf{l}^{\leq j}$, $j = 1, \ldots, i-1$ anyway). We show now that in this case the more general conditions of section 4.4 reduce to simple stability.

First we need some regularity assumptions.

Assumption 1. We assume $\text{pa}_{\mathbf{s}}(A_i) \subset \text{pa}_o(A_i)$ for all $i = 1, \ldots, N$.

The assumption means that the parents of A_i when we follow the strategy $\mathbf{s}$ are a subset of its parents under the observational regime. This can easily be satisfied by redefining, if necessary, $\text{pa}_o(A_i)$ as $\text{pa}_o(A_i) \cup \text{pa}_s(A_i)$, with any added parents having no effect on the conditional probabilities for A_i under o. Note that adding edges from *observable* nodes into action nodes cannot destroy identifiability.

Assumption 2. Each $L_1, \ldots, L_N$ is an ancestor of Y in the graph D_0, where the parents of A_i are as under strategy $\mathbf{s}$, $i = 1, \ldots, N$.
This means that the covariates predict Y when we follow the strategy $\mathbf{s}$.

Remark. Assumption 2 is implied if
(i) each $A_1, \ldots, A_N$ is an ancestor of Y in D_0, *and*

(ii) each $L_1, \ldots, L_N$ is an ancestor of some A_j in D_0.

As we want to investigate whether the actions affect Y, part (i) will be plausible because otherwise we have at least one action of which we know *a priori* that it does not predict Y and this would not typically be included in the investigation. Part (ii) is relevant in the context of optimal strategies as these must in principle be allowed to depend on all previous observable covariates.

Theorem 2. Suppose Assumptions 1 and 2 hold. Then if the graphical check of (5) succeeds, the problem exibits simple stability with respect to $(\mathbf{A}, \mathbf{L}, Y)$.
Proof: see appendix.

In consequence we can say that if the target is to find an optimal strategy and we are asking whether this is possible from a given data situation we do not need to apply the more complex graphical check of (5) but can just check simple stability, which has the advantage that it can be seen from a single graph like e.g. Figure 2(b). This also implies that a sufficient criterion for identification of an optimal strategy is that the causal effects of each A_i on later covariates L_j, $j > i$, can be identified by the back–door criterion as mentioned in section 4.5.

6 RELATION TO OTHER APPROACHES

It has been argued that conditional strategies can be identified if *conditional intervention distributions* can be identified (Pearl, 2000, section 4.2). For the case of a non–stochastic conditional strategy $\mathbf{s}$ that fixes $\mathbf{a} = \mathbf{s}(\mathbf{l})$ this is seen as follows. We have that

$$p(y; \mathbf{s}) = \sum_{\mathbf{l}} p(y|\mathbf{l}; \mathbf{s}) p(\mathbf{l}; \mathbf{s}) \tag{8}$$

(the recursive version of which is based on (2)). As $\mathbf{l}$ in $p(y|\mathbf{l}; \mathbf{s})$ is given, we have $p(y|\mathbf{l}; \mathbf{s}) = p(y|\mathbf{l}; \mathbf{a})$, where $\sigma = \mathbf{a}$ denotes an unconditional strategy and $\mathbf{a} = \mathbf{s}(\mathbf{l})$. Hence we can identify $p(y; \mathbf{s})$ if we can identify $p(y|\mathbf{l}; \mathbf{a})$ for every sequence $\mathbf{l}$ and every unconditional strategy $\sigma = \mathbf{a}$. Shpitser & Pearl (2006) give a sound and complete algorithm to identify such conditional intervention distributions, like $p(y|\mathbf{l}; \mathbf{a})$, which outputs FAIL iff the problem is not identified. This takes a semi–Markovian graph as input which is based on a causal model.

However, for the whole $p(y; \mathbf{s})$ to be identifiable by (8) we also need $p(\mathbf{l}; \mathbf{s})$ to be identifiable, where it is important to note that due to the sequential nature of the problem some covariates will be affected by earlier actions and hence we cannot assume $p(\mathbf{l}; \mathbf{s}) = p(\mathbf{l}; o)$. Tian (2004) gives an example where $p(y|\mathbf{l}; \mathbf{s})$ is identified while $p(\mathbf{l}; \mathbf{s})$ is not. Hence, identifiability of conditional sequential plans is not covered by the identifiability of conditional interventional distributions alone.

Our results do not provide a necessary criterion for identifiability. However, they do not require a semi–Markovian graph nor a causal model (which in our case would assume that we can intervene in any of $L_1, \ldots, L_N$ as well as in $\mathbf{A}$) as long as the background knowledge can be encoded in a DAG on $(\mathbf{A}, \mathbf{L}, \mathbf{U}, Y, \sigma)$. More importantly, as mentioned earlier, simple stability implies that the effect of each action on later covariates $p(\mathbf{l}; \mathbf{s})$ as well as the conditional intervention distribution $p(y|\mathbf{l}; \mathbf{s})$ are identifiable, so that (8) applies.

7 DISCUSSION

We have addressed the question of identifying optimal sequential strategies within the framework of decision theory. Simple stability (3) provides a straightforward graphical check for identifiability on a single influence diagram, and the more involved check for the conditions in section 4.4 that extends Pearl & Robins (1995) approach is in fact not more general.

We would like to point out that even though the target of inference $E(k(Y); \mathbf{s})$ can be constructed using the G–recursion, it is in practice not advisable to estimate an optimal strategy (when it *is* identified) by estimating the individual factors of the G–recursion formula (Robins & Wasserman, 1997). This has motivated the alternative approaches such as suggested by Murphy (2003) and Robins (2004). The reasoning regarding identifiability, however, remains valid.

References

Cowell, R.G., Dawid, A.P., Lauritzen, S.L., Spiegelhalter, D.J. (1999). Probabilistic Networks and Expert Systems. Springer.

Dawid, A.P. (1979). Conditional independence in statistical theory (with discussion). JRSSB, **41**, pp. 1-31.

Dawid, A.P. (2002). Influence diagrams for causal modelling and inference. *Int. Statist. Rev.*, **70**, pp. 161-89.

Dawid, A.P. and Didelez, V. (2005). Identifying the consequences of dynamic treatment strategies. Research Report No. 262, Department of Statistical Science, University College London.

Didelez, V., Dawid, A.P., Geneletti, S. (2006). Direct and indirect effects of sequential decisions. In:

Proc. of the 22th Conference on Uncertainty in Artificial Intelligence (UAI-06), 138-146. AUAI Press.

Lauritzen, S.L. (2001). Causal inference from graphical models. In: O.E. Barndorff-Nielsen, D.R. Cox and C. Klüppelberg (eds.), Complex Stochastic Systems, pp. 63-107. Chapman and Hall.

Moodie, E.E.M., Richardson, T.S., Stephens, D.A. (2007). Demystifying optimal dynamic treatment regimes. Biometrics, **63**, pp. 447-455.

Murphy, S.A. (2003). Optimal Dynamic Treatment Regimes (with discussion), JRSSB, **65**, pp. 331-366.

Pearl, J. (1993). Graphical models, causality and interventions. *Statistical Science*, **8**, pp. 266-9.

Pearl, J. (1995). Causal diagrams for empirical research. *Biometrika*, **82**, pp. 669-710.

Pearl, J. (2000). *Causality — Models, Reasoning and Inference*. Cambridge University Press.

Pearl, J. and Robins, J. (1995). Probabilistic evaluation of sequential plans from causal models with hidden variables. In: Proc. of the 11th Conference on Uncertainty in Artificial Intelligence (UAI-95), pp. 444-453. Morgan Kaufmann.

Raiffa, H. (1968). Decision analysis. Addison–Wesley.

Robins, J. (1986). A new approach to causal inference in mortality studies with sustained exposure periods — application to control for the healthy worker survivor effect. *Mathematical Modelling*, **7**, 1393-512.

Robins, J.M. (1997). Causal inference from complex longitudinal data. In: M. Berkane (ed.), Latent Variable Modeling and Applications to Causality, Lecture Notes in Statistics, Vol. 120, pp. 69-117. Springer.

Robins, J.M. (2004). Optimal structural nested models for optimal sequential decisions. In: D.Y. Lin and P. Heagerty (eds.), Proc. of 2nd Seattle Symposium on Biostatistics. Springer.

Robins, J.M., Wasserman, L. (1997). Estimation of effects of sequential treatments by reparameterizing directed acyclic graphs. In: Proc. of the 13th Conference on Uncertainty in Artificial Intelligence (UAI-97), pp. 444-453. Morgan Kaufmann.

Rosthoj S., Fullwood C., Henderson R., Stewart S. (2008). Estimation of optimal dynamic anticoagulation regimes from observational data: a regret-based approach. Statistics in Medicine, to appear.

Tian, J. (2004). Identifying conditional causal effects. In: Proc. of the 20th Conference on Uncertainty in Artificial Intelligence (UAI-04), pp. 561-568. AUAI Press.

Shpitser, I., Pearl, J. (2006). Identification of conditional interventional distributions. In: Proc. of the 22th Conference on Uncertainty in Artificial Intelligence (UAI-06), pp. 437-444. AUAI Press.

Verma, T., Pearl, J. (1990). Causal Networks: Semantics and Expressiveness. In: Proc. of the 4th Conference on Uncertainty in Artificial Intelligence (UAI-90), pp. 561-568. Elsevier Science.

APPENDIX

PROOF OF THEOREM 1

This is a special case of a more general result (Dawid and Didelez, 2005, § 7.1). The following argument is specialised to the current context, and assumes that events conditioned on have positive probability.

Similar to section 3, define

$$\begin{aligned} f(\mathbf{a}^{\leq i},\mathbf{l}^{\leq i}) &= E_i\{k(Y) \mid \mathbf{a}^{\leq i},\mathbf{l}^{\leq i}\} \\ f(\mathbf{a}^{<i},\mathbf{l}^{\leq i}) &= E_{i-1}\{k(Y) \mid \mathbf{a}^{<i},\mathbf{l}^{\leq i}\}, \end{aligned}$$

where E_i denotes expectation under distribution p_i. In particular, $f(\emptyset) = E\{k(Y)|\mathbf{s}\}$, which is what we wish to compute; while $f(\mathbf{a}^{\leq N},\mathbf{l}^{\leq N}) = E\{k(Y)|\mathbf{a}^{\leq N},\mathbf{l}^{\leq N};o\}$ is estimable from the observational data.

By extended stability, the distribution of (L_i, U_i) given $(\mathbf{A}^{<i},\mathbf{L}^{<i},\mathbf{U}^{<i})$ is the same under both the observational regime o and the strategy $\mathbf{s}$. It then follows from the definition of p_{i-1} that the joint distribution of $(\mathbf{A}^{<i},\mathbf{L}^{\leq i},\mathbf{U}^{\leq i})$ is the same under p_{i-1} as in o, whence in particular $p_{i-1}(l_i|\mathbf{a}^{<i},\mathbf{l}^{<i}) = p(l_i|\mathbf{a}^{<i},\mathbf{l}^{<i};o)$. Thus, $f(\mathbf{a}^{<i},\mathbf{l}^{<i}) =$

$$\begin{aligned} & \sum_{l_i} p_{i-1}(l_i|\mathbf{a}^{<i},\mathbf{l}^{<i}) \times E_{i-1}\{k(Y) \mid \mathbf{a}^{<i},\mathbf{l}^{\leq i}\} \\ = & \sum_{l_i} p(l_i|\mathbf{a}^{<i},\mathbf{l}^{<i};o) \times f(\mathbf{a}^{<i},\mathbf{l}^{\leq i}). \end{aligned} \quad (9)$$

Also, from the construction of p_{i-1}, the fact that $p(a_i|\mathbf{a}^{<i},\mathbf{l}^{\leq i};\mathbf{s})$ is fully determined independently of $\mathbf{u}^{\leq i}$, and (5), $f(\mathbf{a}^{<i},\mathbf{l}^{\leq i}) =$

$$\sum_{a_i} p_{i-1}(a_i \mid \mathbf{a}^{<i},\mathbf{l}^{\leq i}) \times E_{i-1}\{k(Y) \mid \mathbf{a}^{\leq i},\mathbf{l}^{\leq i}\}$$

$$= \sum_{a_i} p(a_i \mid \mathbf{a}^{<i}, \mathbf{l}^{\leq i}; \mathbf{s}) \times f(\mathbf{a}^{\leq i}, \mathbf{l}^{\leq i}), \quad (10)$$

where $E_{i-1}\{k(Y)|\mathbf{a}^{\leq i}, \mathbf{l}^{\leq i}\} = f(\mathbf{a}^{\leq i}, \mathbf{l}^{\leq i})$ due to (5). Together, (9) and (10) show that f can be computed by G-recursion: starting with $f(\mathbf{a}^{\leq N}, \mathbf{l}^{\leq N}) = E\{k(Y) \mid \mathbf{a}^{\leq N}, \mathbf{l}^{\leq N}; o\}$, we will exit the recursion with $f(\emptyset) = E\{k(Y) \mid \mathbf{s}\}$.

PROOF OF THEOREM 2

We introduce the following abbreviations for certain graph-separation properties:

$$\begin{aligned}
\gamma_i &: [Y \perp\!\!\!\perp \sigma | (\mathbf{L}^{\leq i}, \mathbf{A}^{\leq i})]_{D_i} \\
\alpha_{ij} &: [L_{j+1} \perp\!\!\!\perp_i \sigma | (\mathbf{L}^{\leq i}, \mathbf{A}^{\leq i})]_{D_i} \\
\alpha_k &: \alpha_{ij} \text{ holds for all } 0 \leq i \leq j \leq k \\
\beta_k &: [L_{k+1} \perp\!\!\!\perp_\sigma | (\mathbf{A}^{\leq k}, \mathbf{L}^{\leq k})]_D \\
H_k &: \alpha_k \Rightarrow \beta_k
\end{aligned}$$

Also, we use $\text{man}_i(A, B, C)$ to denote the moralised subgraph of D_i on the set $\text{An}(A \cup B \cup C)$[3].

Lemma 1. Suppose Assumptions 1 and 2 hold. Then for $1 \leq i \leq j \leq N$, $\gamma_i \Rightarrow \alpha_{ij}$.

Proof: Fix i, and suppose that for some $j \geq i$, α_{ij} fails. Then there is a path π_1 in $\text{man}_i(\mathbf{A}^{\leq i}, \mathbf{L}^{\leq i}, L_{j+1})$ joining σ to some $L \in L_{j+1}$ avoiding $(\mathbf{A}^{\leq i}, \mathbf{L}^{\leq i})$. Since by Assumption 2 we have L_{j+1} is an ancestor of Y in D_i, π_1 is a path in $\text{man}_i(\mathbf{A}^{\leq i}, \mathbf{L}^{\leq i}, Y)$ with the same property. But again by Assumption 2, there is a descending path π_2 in D_i joining L to Y avoiding $(\mathbf{A}^{\leq i}, \mathbf{L}^{\leq i})$ since these are non–descendants of L_{j+1}. Hence, the concatenation of π_1 and π_2 is a path in $\text{man}_i(\mathbf{A}^{\leq i}, \mathbf{L}^{\leq i}, Y)$ joining σ and Y while avoiding $(\mathbf{A}^{\leq i}, \mathbf{L}^{\leq i})$ showing that γ_i fails.

Lemma 2. Suppose Assumption 1 holds. Then H_k holds, for all $0 \leq k \leq N$.

Proof: The proof is by induction on k. H_0 holds since β_0 holds by extended stability. Now fix $k \leq N$ and suppose that H_l holds for all $0 \leq l < k$ and argue that H_k holds. To do so we assume that α_k holds, and argue that β_k follows.

Since for all $0 \leq l < k$ α_l must hold, from H_l so must β_l.

Before proceeding we introduce some notation. For $1 \leq j \leq k$, we denote by M_j (resp. N_j) the moralised ancestral graph of $(\mathbf{A}^{\leq j}, \mathbf{L}^{\leq j}, L_{k+1})$ in D (resp. in D_j). We write $V_1 \searrow V_2$ to denote that there is a descending directed path in D from node V_1 to node V_2 whose intermediate nodes are all in $\mathbf{U}$.

Now suppose β_k fails, then there exists a path π in M_k having the following property:

P: π connects σ to L_{k+1}, all intermediate nodes being in $\mathbf{U}$.

Let Π_j denote the property that any moral link in π is due to a common child in C in $(\mathbf{A}^{\leq j}, \mathbf{L}^{\leq j})$.

If π contains a moral link due to a common child $C \notin (\mathbf{A}^{\leq k}, \mathbf{L}^{\leq k})$ then $C \in \mathbf{U}$, and we can modify π by adding the intermediate node C and removing the initial moral link, while still satisfying property **P**. Proceeding in this way for all such moral links we see that we can suppose that Π_k holds.

Let $U_1 \in \mathbf{U}$ be the first internal node in π after σ: this must exist because L_{k+1} and σ are not directly connected, nor can they have a common child in M_k. The link σ—U_1 can only be a moral link due to a common child A_i for some $i \leq k$. Then we cannot have $U_1 \searrow L_{k+1}$ since this would create a path violating α_{ik}.

Let U_2 be the first internal node, if one exists, in π that is not an ancestor of $(\mathbf{A}^{\leq k}, \mathbf{L}^{\leq k})$ in D. Then we must have $U_2 \searrow L_{k+1}$ by some path π_2, say. In particular $U_1 \neq U_2$. Denote by π_1 the section of π between σ and U_2, and let π_1^o be this section stripped of its endpoints. Since $U_1 \in \pi_1^o$, it is not empty. Now replace the original section section of π from U_2 to L_{k+1} by π_2. The new path will still possess properties **P** and Π_k. We replace π by the concatenation of π_1 and π_2 which will be renamed π for the sequel.

Consider now the hypotheses (Q_j), $0 \leq j \leq k$, defined by $Q_j : \pi_1^o \subset \text{an}(\mathbf{A}^{\leq j}, \mathbf{L}^{\leq j})$ and property Π_j holds. We shall show that $Q_j \Rightarrow Q_{j-1}$.

Thus suppose Q_j. There cannot exist $U \in \pi_1^o$ with $U \searrow A_j$ since this would violate α_{jk}. We deduce that $\pi_1^o \subset \text{an}(\mathbf{A}^{\leq j}, \mathbf{L}^{\leq j})$ and any moral link in π must be due to a common child in $(\mathbf{A}^{<j}, \mathbf{L}^{\leq j})$.

Now Q_j implies that π is a path in M_j. Then there cannot exist $U \in \pi_1^o$ with $U \searrow L_j$ since this would violate β_{j-1}. We deduce Q_{j-1}.

Now Q_k holds by construction. Applying the above repeatedly we deduce Q_0, which can only hold if π_1^o is empty. But $U_1 \in \pi_1^o$. This contradiction proves that β_k holds and so we have demonstrated H_k. The desired result follows by induction.

Proof of Theorem 2: The graphical check succeeds just when we can show γ_i, $1 \leq i \leq N$. By the above Lemmas, α_N must hold which implies that β_j, $1 \leq j \leq N$ does. This is exactly the condition for simple stability.

[3]We use the moralisation criterion here to verify graph separation in DAGs

Strategy Selection in Influence Diagrams using Imprecise Probabilities

Cassio P. de Campos
Electrical, Computer and Systems Eng. Dept.
Rensselaer Polytechnic Institute
Troy, NY, USA
decamc@rpi.edu

Qiang Ji
Electrical, Computer and Systems Eng. Dept.
Rensselaer Polytechnic Institute
Troy, NY, USA
jiq@rpi.edu

Abstract

This paper describes a new algorithm to solve the decision making problem in Influence Diagrams based on algorithms for credal networks. Decision nodes are associated to imprecise probability distributions and a reformulation is introduced that finds the global maximum strategy with respect to the expected utility. We work with Limited Memory Influence Diagrams, which generalize most Influence Diagram proposals and handle simultaneous decisions. Besides the global optimum method, we explore an anytime approximate solution with a guaranteed maximum error and show that imprecise probabilities are handled in a straightforward way. Complexity issues and experiments with random diagrams and an effects-based military planning problem are discussed.

1 INTRODUCTION

An influence diagram is a graphical model for decision making under uncertainty [13]. It is composed by a directed graph where utility nodes are associated to profits and costs of actions, chance nodes represent uncertainties and dependencies in the domain and decision nodes represent actions to be taken. Given an influence diagram, a strategy defines which decision to take at each node, given the information available at that moment. Each strategy has a corresponding expected utility. One of the most important problems in influence diagrams is *strategy selection*, where we need to find the strategy with maximum expected utility. A simple approach is to evaluate each possible strategy and compare their expected utilities. However, the number of strategies grows exponentially in the number of decision to be taken.

In this paper, we propose a new idea to find the best strategy based on a reformulation of the problem as an inference in a credal network [4]. We show through experiments that this approach can handle small and medium diagrams exactly, and provides an anytime approximation in case we stop the process early. Our idea works with a very general class of influence diagrams, named *Limited Memory Influence Diagrams* (LIMIDs) [15]. *Limited Memory* means that the assumption of *no-forgetting* usually employed in Influence Diagrams (that is, values of observed variables and decisions that have been taken are remembered at all later times) is relaxed. This class of diagrams is interesting because most other influence diagram proposals can be efficiently converted into LIMIDs.

To solve strategy selection, many approaches work on special cases of influence diagrams, exploiting their characteristics to improve performance. In many cases, it is assumed that there is an ordering on which the decisions are to be taken and the no-forgetting rule, so as previous decisions are assumed to be known in the moment of the current decision [14, 18, 19, 20, 21]. The ordering of decision nodes is exploited to evaluate the optimal strategy. There are also proposals in the class of simultaneous influence diagrams, where decisions are assumed to have no antecedents. This assumption reduces the number of possible strategies and allows for factorization ideas [22]. LIMIDs do not have assumptions about no-forgetting and ordering for decisions, even though it is possible to convert diagrams that have such assumptions into LIMIDs.

In order to test our method, we generate a data set of random influence diagrams. Empirical results indicate that the accuracy of our method is better than other approaches'. We also apply our idea to solve an Effects-based operations (EBO) military planning. The EBO approach seeks for a campaign objective by considering direct, indirect and cascading effects of military, diplomatic, psychological and economic actions [6, 11]. We use an influence diagram to model an EBO hypothetical problem.

Section 2 introduces our notation for influence diagrams and the problem of strategy selection. Section 3 describes the framework of credal networks and the inference problem on such networks. Section 4 presents how we solve strategy selection through a reformulation of the problem as an inference in credal networks. Section 5 presents some experiments, including the EBO military planning problem, and finally Section 6 concludes the paper and indicates future work.

2 INFLUENCE DIAGRAMS

A Limited Memory Influence Diagram $\mathcal{I}$ is composed by a directed acyclic graph $(\mathcal{V}, E)$ where nodes are partitioned in three types: chance, decision and utility nodes. Let $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{U}$ be the set of chance, decision and utility nodes, respectively, and let $\mathcal{X} = \mathcal{C} \cup \mathcal{D}$. Links of E characterize dependencies among nodes. Explicitly, links toward a chance node indicate probabilistic dependence of the node on its parents; links toward a decision node indicate which information is available to take such decision, and links toward utility nodes represent that an utility for those parents is to be considered (utility nodes may not have children). Associated to each node, there are some parameters:

1. A *chance node* has an associated categorical random variable C with finite domain Ω_C and conditional probability distributions $p(C|\pi_j(C))$, for each configuration $\pi_j(C)$ of its parents $\pi(C)$ in the graph. j is used to indicate a configuration of the parents of C, that is, $\pi_j(C) \in \Omega_{\pi(C)}$, where the notation $\Omega_{\mathcal{V}'} = \times_{V \in \mathcal{V}'} \Omega_V$, for any $\mathcal{V}' \subseteq \mathcal{V}$.

2. A *decision node* D is associated to a finite set of mutually exclusive alternatives Ω_D. Parents of D describe the information that is available at the moment on which decision D has to be taken.

3. An *utility node* U is associated to a rational function $f_U : \Omega_{\pi(U)} \to \mathcal{Q}$. The value corresponding to a parent configuration is the profit (cost is viewed as negative profit) of such parent configuration. Utility nodes have no children.

A simple example is depicted in Figure 1. Decision nodes are represented by rectangles, chance nodes by ellipses and utility nodes by diamonds. *do_ground_attack* has an associated cost, which is depicted by the corresponding utility node. The same is modeled for *bomb_bridge*. The goal is to achieve *territory_occupation*, which also has an utility (the profit of the goal). *ground_attack* and *bridge_condition* represent the uncertain outcomes of the corresponding actions. Note that there is no known ordering on which

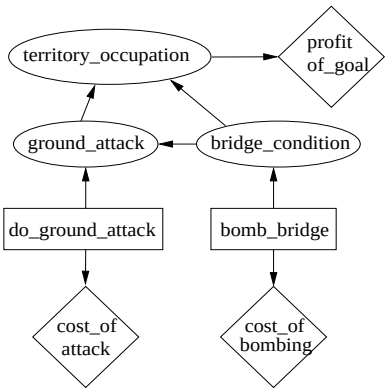


Figure 1: Simple Influence Diagram example.

decisions must be taken. Although decision nodes have no parents in this example, there is no such restriction.

A *policy* δ_D for the decision node D is a function $\delta_D : \Omega_{D \cup \pi(D)} \to [0, 1]$ defined for each alternative of D and each configuration of $\pi(D)$ such that, for each $\pi_j(D) \in \Omega_{\pi(D)}$ we have $\sum_{d \in \Omega_D} \delta_D(d, \pi_j(D)) = 1$. A *pure policy* is a policy such that its image is integer ($\delta_D : \Omega_{D \cup \pi(D)} \to \{0, 1\}$), and thus specifies with certainty which action (alternative of D) is taken for each parent configuration (in a pure policy, only one $\delta_D(d, \pi_j(D))$ for each $\pi_j(D)$ will be non-zero as they sum 1). A *strategy* Δ is a set of policies $\{\delta_D : D \in \mathcal{D}\}$, one for each decision node of the diagram. A *pure strategy* is composed only by pure policies.

The expected utility $EU(\Delta)$ of a strategy Δ is evaluated through the following equation:

$$\sum_{\mathbf{x} \in \Omega_{\mathcal{X}}} \left(\prod_C p(x_C|\pi_j(C)) \prod_D \delta_D(x_D) \sum_U f_U(\pi_{j'}(U)) \right), \tag{1}$$

where x_C, $\pi_j(C)$, x_D and $\pi_{j'}(U)$ are respectively the projections of $\mathbf{x}$ in Ω_C, $\Omega_{\pi(C)}$, $\Omega_{D \cup \pi(D)}$ and $\Omega_{\pi(U)}$. This equation means that, given a strategy, its expected utility is the sum of the utility values weighted by the probability of each diagram configuration (for all configurations). The maximum expected utility is obtained over all possible strategies:

$$MEU = \max_{\Delta} EU(\Delta).$$

The problem of *strategy selection* is to obtain the strategy that maximizes its expected utility, that is, $\operatorname{argmax} \max_{\Delta} EU(\Delta)$.

3 CREDAL NETWORKS

We need some concepts of credal networks before presenting the reformulation to solve strategy selection. A convex set of probability distributions is called a

credal set [4]. A credal set for X is denoted by $K(X)$; we assume that every random variable is categorical and that every credal set has a finite number of vertices. Given a credal set $K(X)$ and an event A, the *upper* and *lower* probability of A are respectively $\max_{p(X)\in K(X)} p(A)$ and $\min_{p(X)\in K(X)} p(A)$. A conditional credal set is a set of conditional distributions, obtained by applying Bayes rule to each distribution in a credal set of joint distributions.

A (separately specified) *credal network* $N = (G, \mathbb{X}, \mathbb{K})$ is composed by a directed acyclic graph $G = (V, E)$ where each node of V is associated with a random variable $X_i \in \mathbb{X}$ and with a collection of conditional credal sets $K(X_i|\pi(X_i)) \in \mathbb{K}$, where $\pi(X_i)$ denotes the parents of X_i in the graph. Note that we have a conditional credal set related to X_i for each configuration $\pi_j(X_i) \in \Omega_{\pi(X_i)}$. A root node is associated with a single marginal credal set. We take that in a credal network every random variable is independent of its non-descendants non-parents given its parents; this is the *Markov condition* on the network. In this paper we adopt the concept of *strong independence*[1]: two random variables X_i and X_j are strongly independent when every extreme point of $K(X_i, X_j)$ satisfies standard stochastic independence of X_i and X_j (that is, $p(X_i|X_j) = p(X_i)$ and $p(X_j|X_i) = p(X_j)$) [4]. Strong independence is the most commonly adopted concept of independence for credal sets, probably due to its connection with standard stochastic independence.

Given a credal network, its *extension* is any joint credal set that satisfies all constraints encoded in the network. The *strong extension* $\mathcal{K}$ of a credal network is the largest joint credal set such that every variable is strongly independent of its non-descendants non-parents given its parents. The strong extension of a credal network is the joint credal set that contains every possible combination of vertices for all credal sets in the network [5]; that is, each vertex of a strong extension factorizes as follows:

$$p(X_1, \ldots, X_n) = \prod_i p(X_i|\pi(X_i)) \,. \qquad (2)$$

Thus, a credal network can be viewed as a representation for a set of Bayesian networks with distinct parameters but sharing the same graph.

3.1 INFERENCE

A *marginal inference* in a credal network is the computation of upper (or lower) probabilities in an extension of the network. If X_q is a *query* variable, then a marginal inference is the computation of tight bounds for $p(x_q)$ for one or more categories x_q of X_q. For inferences in strong extensions, it is known that distributions that maximize $p(x_q)$ belong to the set of vertices of the extension [12]. So, an inference can be produced by combinatorial optimization, as we must find a vertex for each local credal set $K(X_i|\pi(X_i))$ so that Expression (2) leads to a maximum of $p(x_q)$. In general, inference offers tremendous computational challenges, and exact inference algorithms based on enumeration of all potential vertices face serious difficulties [4].

A different way to solve the problem is to recognize that an upper (or lower) value for $p(x_q)$ may be obtained by the optimization of a multilinear polynomial over probability values, subject to constraints. This idea is discussed in the literature and different methods to reformulate the inference problem were proposed [7, 9]. Empirical results suggest that this is the most effective way for exact inferences. In the next section, we describe an idea based on bilinear programming [9] to perform inferences in credal networks and show how it can be employed to solve the strategy selection problem of influence diagrams.

4 STRATEGY SELECTION AS A CREDAL NET INFERENCE

Suppose we want to find the strategy Δ_{opt} that maximizes the expected utility in an influence diagram $\mathcal{I}$, that is, $\Delta_{opt} = \operatorname{argmax} MEU$. Let $\underline{f}$ and $\overline{f}$ be the minimum and maximum utility values specified in the diagram for all possible utility nodes and parent configurations, that is,

$$\underline{f} = \min_{U,\pi_j(U)} f_U(\pi_j(U)), \qquad \overline{f} = \max_{U,\pi_j(U)} f_U(\pi_j(U)).$$

We create an identical influence diagram $\mathcal{I}'$ except that the utility function f'_U (for each node U) is defined as

$$\forall \pi_j(U) \quad f'_U(\pi_j(U)) = \frac{f_U(\pi_j(U)) - \underline{f}}{\overline{f} - \underline{f}}.$$

The denominator is positive because $\underline{f} < \overline{f}$ (if $\underline{f} = \overline{f}$, then the influence diagram is trivial as all utility values are equal). We note that this transformation is similar to that proposed by Cooper [2]. It is not hard to see that $\operatorname{argmax} MEU = \operatorname{argmax} MEU'$ (just take the terms out of summations in Equation (1)), and

$$\max_{\Delta} EU'(\Delta) = \frac{\max_{\Delta} EU(\Delta) - |\mathcal{U}|\underline{f}}{\overline{f} - \underline{f}}.$$

This implies that strategy selection in $\mathcal{I}$ is the same as strategy selection in $\mathcal{I}'$. Now, we translate the selection problem of $\mathcal{I}'$ to a credal network inference. Suppose we define a credal network with a similar graph as $\mathcal{I}'$ such that:

[1] We note that other concepts of independence are found in the literature [3, 10].

- Chance nodes are directly translated as nodes of the credal network (parents are the same as in $\mathcal{I}'$).
- Utility nodes are translated to binary random nodes. Let U be an utility node with function f_U. In the credal network, U becomes a binary node (with the same parents as before) and categories u and $\neg u$ such that: $p(u|\pi_j(U)) = f_U(\pi_j(U))$ and $p(\neg u|\pi_j(U)) = 1 - p(u|\pi_j(U))$ [2].
- Decision nodes are translated to probabilistic nodes with imprecise distributions such that policies become probability distributions (in fact, according to our definition of policy, they are already greater than zero and sum 1). Thus, $p(d|\pi_j(D)) = \delta_D(d, \pi_j(D))$ for all d and $\pi_j(D)$. Note that $p(D|\pi_j(D))$, for each $\pi_j(D)$, is a distribution with unknown probability values (this interpretation of decision nodes as imprecise probability nodes is discussed by Antonucci and Zaffalon, see e.g. [1]).

Using this credal network formulation, the expected utility of a strategy Δ can be written as

$$EU'(\Delta) = \sum_{\mathbf{x}\in\Omega_{\mathcal{X}}} \left(\prod_X p_\Delta(x|\pi_j(X)) \sum_U p(u|\pi_{j'}(U)) \right),$$

where x, $\pi_j(X)$ and $\pi_{j'}(U)$ are projections of $\mathbf{x}$ into the corresponding domains, X ranges on all nodes corresponding to chance and decision nodes of the influence diagram, and p_Δ represents the distribution induced by the strategy Δ, that is, when the strategy is chosen, p_Δ is a known probability distribution.

With some simple manipulations, we have:

$$EU'(\Delta) = \sum_{\mathbf{x}\in\Omega_X} \left(p_\Delta(\mathbf{x}) \sum_U p(u|\pi_{j'}(U)) \right),$$

$$EU'(\Delta) = \sum_{\mathbf{x}\in\Omega_X} \left(\sum_U p(u|\pi_{j'}(U)) p_\Delta(\mathbf{x}) \right),$$

$$EU'(\Delta) = \sum_U \sum_{\mathbf{x}\in\Omega_X} p_\Delta(u, \mathbf{x}) = \sum_U p_\Delta(u),$$

and then

$$MEU' = \max_\Delta \sum_U p_\Delta(u) = \max_{p\in\mathcal{K}} \sum_U p(u),$$

where $p \in \mathcal{K}$ means that we select a distribution p in the extension of the credal network. In fact the only places p may vary are related to the imprecise probabilities of the former decision nodes. When we select p, we get a precise distribution that has a corresponding strategy Δ. So, we have a credal network and need to find a distribution p that maximizes the sum of marginal probabilities of the U nodes.

4.1 INFERENCE AS AN OPTIMIZATION PROBLEM

The sum of marginal inferences in the credal network can be formulated as a multilinear programming problem. The goal is to maximize the expression

$$\sum_U p(u) = \sum_U \sum_{\mathbf{x}\in\Omega_{\mathcal{X}}} \left(p(u|\pi_{j'}(U)) \prod_X p(x|\pi_j(X)) \right), \tag{3}$$

where x, $\pi_{j'}(U)$ and $\pi_j(X)$ are the projections of $\mathbf{x}$ in the corresponding domains, and where some distributions $p(X|\pi_j(X))$ are precisely known and others are imprecise. In this formulation we must deal with a large number of multilinear terms. To avoid them, we briefly describe the bilinear transformation procedure proposed by de Campos and Cozman [9] to replace the large Expression (3) by simple bilinear expressions. We refer to [9] for additional details.

The idea is based on a *precedence ordering* of the network variables, which is an ordering where all ancestors of a given variable in the network's graph appear before it in the ordering. The bilinear transformation algorithm processes the network variables top-down: at each step some constraints are generated that define the relationship between the query and the current variable being processed. A variable may be processed only if all its ancestors have already been processed. The active nodes at each step form a path-decomposition of the network's graph.

To better explain the method, we take the example of Figure 1. For simplicity, assume that variables are binary[2] (with categories b and $\neg b$) renamed as follows: *do_ground_attack* is D_1, *bomb_bridge* is D_2, *cost_of_attack* is U_1, *cost_of_bombing* is U_2, *ground_attack* is C_1, *bridge_condition* is C_2, *territory_occupation* is C_3, and finally *profit_of_goal* is U_3.

After the translation of the utility functions into probability distributions and the replacement of decision nodes by nodes with imprecise probabilities (as previously described), we have a credal network and need to maximize the sum of the marginal probabilities of the U nodes. In fact this is an extension of the standard query in a credal network, because we have a summation instead of a single probability to maximize. So the objective function is max $p(u_1) + p(u_2) + p(u_3)$ (there are three utility nodes in the example) subject to constraints that define each marginal probability $p(u_1)$, $p(u_2)$ and $p(u_3)$. To create these constraints, we run a symbolic inference based on the precedence ordering for each of the marginal probabilities. The constraints for $p(u_1)$ and $p(u_2)$ are very

[2]The method works on non-binary variables as well. The assumption is made here for ease of expose.

simple: $p(u_1) = p(u_1|d_1)p(d_1) + p(u_1|\neg d_1)p(\neg d_1)$ and $p(u_2) = p(u_2|d_2)p(d_2)+p(u_2|\neg d_2)p(\neg d_2)$, because they only depend on one other variable. Note that $p(d_1)$, $p(\neg d_1)$, $p(d_2)$, and $p(\neg d_2)$ that appear in these constraints are unknown and thus become optimization variables in the bilinear problem.

To write the constraints for $p(u_3)$, we need to choose a precedence ordering. We will use the ordering $D_2, C_2, D_1, C_1, C_3, U_3$ (variables U_1 and U_2 do not appear in the order as they are not relevant to evaluate the marginal $p(u_3)$). Hence, the first variable to be processed is D_2. We write a constraint that relates the query u_3 and probabilities $p(D_2)$ (which are defined in the network specification):

$$p(u_3) = \sum_{d \in \{d_2, \neg d_2\}} p(d) \cdot p(u_3|d).$$

D_2 now appears in the conditional part of $p(u_3|d)$, which may be viewed as an artificial term in the optimization, as it does not appear in the network. Because of that, we must create constraints to define $p(u_3|d)$ in terms of network parameters (for all categories $d \in D_2$). According to our chosen ordering, the current variable to be processed is C_2. Thus,

$$\begin{aligned} p(u_3|d_2) &= \sum_{c \in \{c_2, \neg c_2\}} p(c|d_2) \cdot p(u_3|c), \\ p(u_3|\neg d_2) &= \sum_{c \in \{c_2, \neg c_2\}} p(c|\neg d_2) \cdot p(u_3|c). \end{aligned}$$

Note that $p(u_3|c) = p(u_3|c, d)$ (for any d), so we use the simpler. At this stage, our query is conditioned on C_2. Following the same idea, we process D_1, obtaining

$$\begin{aligned} p(u_3|c_2) &= \sum_{d \in \{d_1, \neg d_1\}} p(d) \cdot p(u_3|c_2, d), \\ p(u_3|\neg c_2) &= \sum_{d \in \{d_1, \neg d_1\}} p(d) \cdot p(u_3|\neg c_2, d). \end{aligned}$$

Now the current variable to be treated is C_1, and our query is conditioned on C_2, D_1, that is, we must define how to evaluate $p(u_3|C_2, D_1)$ for all configurations. Thus, for all $c \in \{c_2, \neg c_2\}$ and $d \in \{d_1, \neg d_1\}$:

$$p(u_3|c, d) = \sum_{c' \in \{c_1, \neg c_1\}} p(c'|c, d) \cdot p(u_3|c, c').$$

At this moment, u_3 is conditioned on C_1, C_2 in the artificial term $p(u_3|c, c')$ (D_1 is not present in the artificial term as C_1, C_2 separate u_3 from D_1). Now we process C_3: for all $c' \in \{c_1, \neg c_1\}$ and $c \in \{c_2, \neg c_2\}$

$$p(u_3|c, c') = \sum_{c'' \in \{c_3, \neg c_3\}} p(c''|c, c') \cdot p(u_3|c'').$$

Note that, as $p(u_3|c'')$ is specified in the network, we can stop. All artificial terms are related (through constraints) to parameters of the network. Besides all these constraints, we also include simplex constraints to ensure that probabilities sum 1.

Hence, we have a collection of linear and bilinear constraints on which non-linear programming can be employed [7]. It is also possible to use linear integer programming [9]. The steps to achieve a linear integer programming formulation are simple, because the only non-linear terms of the problem have the format $b \cdot t$, where $b \in \{0, 1\}$ and $t \in [0, 1]$. b is an unknown probability value of the credal network (which is zero or one because the solution we look for lies on extreme points of credal sets [12]) and t is a constant or an artificial term created in the procedure just described. To *linearize* the problem, $b \cdot t$ is replaced by an additional artificial optimization variable y and the following constraints are inserted: $0 \leq y \leq b$ and $t - 1 + b \leq y \leq t$. After replacing all non-linear terms using this idea, the problem becomes a linear integer programming problem, where a solution is also a solution for the strategy selection in the initial influence diagram.

We emphasize that, as we are translating the strategy selection problem into a credal network inference, it is straightforward to use imprecise probabilities in the chance nodes of the influence diagram. Intervals or sets of probabilities may be used. The translation works in the same way, but the generated problem will have more imprecise probabilities to optimize.

The following theorem shows that, when reformulating the strategy selection problem as a modified credal network inference, we are not making use of "more effort" than necessary, that is, strategy selection has the same complexity as inference in credal networks.

Theorem 1 *Let $\mathcal{I}$ be a LIMID and k a rational. Deciding whether there is a strategy Δ such that MEU is greater than k is NP-Complete when $\mathcal{I}$ has bounded induced width,*[3] *and NP^{PP}-Complete in general.*

Proof sketch: Pertinence for the bounded induced width case is achieved because (given a strategy) we can compute *MEU* and verify if it is greater than k in polynomial time (using the reformulation and the sum of marginal queries, each marginal query takes polynomial time in a bounded induced width Bayesian network); in the general case, we can perform this verification using a PP oracle. Hardness for the bounded induced width case is obtained with the same reduc-

[3]The maximum clique and the maximum degree in the moral graph are bounded by a logarithmic function in the size of the input needed to specify the problem, which for instance includes polytrees.

tion as in [8] from the MAXSAT problem (replacing the credal nodes with decision nodes and introducing a single utility node). In the general case, the same reduction as in [17] from E-MAJSAT can be used (MAP nodes are replaced by decision nodes). □

5 EXPERIMENTS

We conduct two experiments with the procedure. First, we use random generated influence diagrams to compare the solutions obtained by our procedure (which we call CR for *credal reformulation*) against the *Single Policy Updating* (SPU) of Lauritzen and Nilsson [15]. Later we work with a practical EBO military planning problem and compare the method against the factorization of Zhang and Ji [22].[4]

Concerning random influence diagrams, we have generated a data set based on the total number of nodes and the number of decision nodes. The configurations chosen are presented in the first two columns of Table 1. We have from 10 to 120 nodes, where 3 to 35 are decision nodes. The number of utility nodes is chosen equal to the number of decision nodes. Each line in Table 1 contains the average result for 30 random generated diagrams within that configuration. The third column of the table shows the approximate average number of distinct strategies in the diagrams that would need to be evaluated by a brute force method.

The three columns of the CR method show the time spent to solve the problem, the number of nodes evaluated in the branch-and-bound tree of the optimization procedure (which is significantly smaller than the total number of strategies in brute force) and the maximum error of the solution (all numbers are averages). After the reformulation, the CPLEX solver [16] is used, which includes a heuristic search before starting the branch-and-bound procedure. The evaluations of this heuristic search are not counted in the fifth column of Table 1. Note that the first five rows are separated from the last three because they strongly differ on the size of the search space (exact solutions were found only for the former). The maximum error of each solution is obtained straightforward from the relaxation of the linear integer problem. The last two columns of Table 1 show the time and maximum error of the SPU approximate procedure. Although very fast, the SPU procedure has worse accuracy than the "approximate" CR (solution was approximate in last three rows because we have imposed a time-limit of ten minutes for each run). Furthermore, SPU does not provide an upper bound for the best possible expected utility, as obtained by CR. Still, a possible improvement is to use SPU to provide an initial guess to the optimization.

[4]The factorization idea only works on simultaneous influence diagrams, so it was not used in the other test cases.

5.1 EBO MILITARY PLANNING

In this section we describe the performance of our method in an hypothetical Effects-based Operations planning problem [11]. An influence diagram similar to the model described by Zhang and Ji [22] is employed. Its graph is shown in Figure 2. The goal is to win a war, which is represented by the *Hypothesis* node (on top of Figure 2). Just below there are the subgoals *Air_superiority*, *Territory_occupation*, and *Commander_surrender*, which are directly related to the main goal. There are eleven decision nodes (represented by rectangles): *destroy_C2* (C2 stands for *Command and Control*), *destroy_Radars*, *destroy_Communications*, *launch_air_strike*, *destroy_RD*, *destroy_storage*, *destroy_assembly*, *launch_ground_attack*, *launch_broadcasting*, *capture_bodyguard*, *use_special_force*. Just above decision nodes, we have chance nodes representing the outcomes of performing such actions (they indicate the workability of such systems), and below we have utility nodes (diamond-shaped nodes) describing the cost of each action. Furthermore, we have six chance nodes (in the center of the figure) indicating general workability of *IADS* (Integrated Air Defense System), *Air_force*, *Artillery*, *Ground_force*, *Morale* and *Commander_in_custody* with respect to enemy forces. The overall profit of winning is given by the node U_H, child of *Hypothesis*.

As this is an hypothetical example, we define utility functions and probability distributions as follows:

- Probability of *Hypothesis* is one given that all subgoals are achieved. If one of subgoals is not achieved, then the probability of *Hypothesis* is 60%; if two of them are not achieved, then the probability of success is 30%; if none of subgoals is achieved, then we certainly fail in the campaign.
- For the subgoals *Air_superiority*, *Territory_occupation*, and *Commander_surrender*, we define that the subgoal is accomplished with probability one when both children were achieved, 50% when only one child is achieved, and zero when none is achieved.
- For the probabilities of *IADS*, *Air_force*, *Artillery*, *Ground_force*, *Morale* and *Commander_in_custody*, we define a decrease of 50% for each unaccomplished child (with a minimum of zero, of course). Any node has probability zero if two or more of its children are not achieved.
- The outcomes of actions (chance nodes above decision nodes) have 90% of success. For exam-

Nodes		Approx.# of	CR			SPU	
Total	Decision	Strategies	Time(sec)	Evals (B&B)	Max.Error(%)	Time(sec)	Max.Error(%)
10	3	2^{17}	0.66	5	0.000	0.10	0.740
20	6	2^{34}	1.73	125	0.000	0.39	2.788
50	10	2^{51}	30.42	4048	0.000	1.62	2.837
60	15	2^{52}	29.77	2937	0.000	2.99	1.964
70	20	2^{54}	125.06	7132	0.000	5.52	3.448
120	25	2^{102}	254.80	15626	0.544	11.58	2.193
120	30	2^{116}	403.13	5617	4.639	13.79	7.281
120	35	2^{120}	578.99	9307	5.983	16.87	11.584

Table 1: Average results on 30 random influence diagrams of different sizes for the CR and SPU methods.

ple, *destroy_Radars* will have *EW/GCI_radars* destroyed with 90% of odds (EW/GCI means *Early Warning/Ground Control Interception*).

- The reward of achieving the main goal is 1000, while not achieving it costs 500.

- Costs of actions are as follows: *ground_attack* is 150, *use_special_force* is 100, *capture_bodyguard* is 80, *air_strike* is 50, and other actions cost 20 each.

For this problem, the best strategy found by SPU has expected utility of -55.2825, and suggests to take all action except *destroy_RD*, *destroy_storage*, *destroy_assembly* and *launch_ground_attack*. The global optimum strategy is found in less than 5 seconds with our method and has expected utility equal to 156.4051 (all actions are taken). This is much faster than the solution reported by [22] (around 45 seconds).

6 CONCLUSION

We discuss in this paper a new idea for strategy selection in Influence Diagrams. We work with the Limited Memory Influence Diagram, as it generalizes many of the influence diagram proposals. The main contribution is the reformulation of the problem as a credal network inference, which makes possible to find the global maximum strategy for small- and medium-sized influence diagrams. Experiments indicate that many instances can be treated exactly. As far as we know, no deep investigation of exact procedures for this class of diagrams has been conducted.

Because of the characteristics of our procedure, an anytime approximate solution with a maximum guaranteed error is available during computations. It is clear that large diagrams must be treated approximately. Nevertheless, in the conducted experiments, our method produced results that surpass existing algorithms. Although spending more time, many situations require a solution to be as good as possible, while time is a secondary issue. The ability of our approach to provide an upper bound for the result is also valuable, which is not available with the SPU method.

We also discuss the theoretical complexity of the problem, which is derived from the known properties of MAP problems in Bayesian networks and belief updating inferences in credal networks. The complexity results show that the proposed idea is not making use of a harder problem to solve a simpler one, as the complexity of strategy selection is the same as the complexity of inferences in credal networks.

Because strategy selection in influence diagrams and inferences in credal networks are related, improvements on algorithms of credal networks can be directly applied to influence diagram problems. The application of other approximate techniques based on credal networks seems a natural path for investigation. We also intend to explore other optimization criteria for influence diagrams with imprecise probabilities, besides expected utility. Proposals in the theory of imprecise probabilities might be applied to this setting.

Acknowledgements

The work described in this paper is supported in part by the U.S. Army Research Office grant W911NF0610331.

References

[1] A. Antonucci and M. Zaffalon. Decision-theoretic specification of credal networks: A unified language for uncertain modeling with sets of Bayesian networks. *Int. J. Approx. Reason.*, in press, doi:10.1016/j.ijar.2008.02.005, 2008.

[2] G. F. Cooper. A method for using belief updating as influence diagrams. In *Conf. on Uncertainty in Artif. Intelligence*, p. 55–63, Minneapolis, 1988.

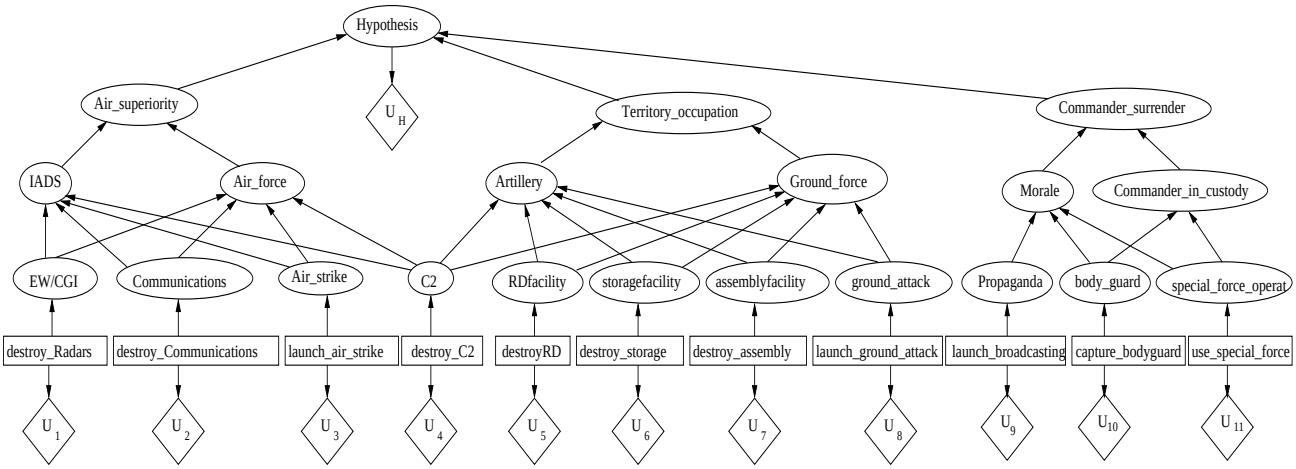


Figure 2: Influence Diagram for an hypothetical EBO-based planning problem.

[3] I. Couso, S. Moral, and P. Walley. A survey of concepts of independence for imprecise probabilities. *Risk, Decision and Policy*, 5:165–181, 2000.

[4] F. G. Cozman. Credal networks. *Artif. Intelligence*, 120:199–233, 2000.

[5] F. G. Cozman. Separation properties of sets of probabilities. In *Conf. on Uncertainty in Artif. Intelligence*, p. 107–115, San Francisco, 2000.

[6] P. Davis. Effects-based operations: a grand challenge for the analytical community. Technical report, Rand corp., 2003. MR1477.

[7] C. P. de Campos and F. G. Cozman. Inference in credal networks using multilinear programming. In *Second Starting AI Researcher Symposium*, p. 50–61, Valencia, 2004. IOS Press.

[8] C. P. de Campos and F. G. Cozman. The inferential complexity of Bayesian and credal networks. In *Int. Joint Conf. on Artif. Intelligence*, p. 1313–1318, 2005.

[9] C. P. de Campos and F. G. Cozman. Inference in credal networks through integer programming. In *Int. Symp. on Imprecise Probability: Theories and Applications*, p. 145–154, 2007.

[10] L. de Campos and S. Moral. Independence concepts for convex sets of probabilities. In *Conf. on Uncertainty in Artif. Intelligence*, p. 108–115, San Francisco, 1995.

[11] D. A. Deptula. Effects-based operations: change in the nature of warfare. *Defense and Airpower Series*, p. 3–6, 2001.

[12] E. Fagiuoli and M. Zaffalon. 2U: An exact interval propagation algorithm for polytrees with binary variables. *Artif. Intelligence*, 106(1):77–107, 1998.

[13] R. A. Howard and J. E. Matheson. *Influence diagrams*, volume II, p. 719–762. Strategic Decisions Group, Menlo Park, 1984.

[14] F. Jensen, F. V. Jensen, and S. L. Dittmer. From influence diagrams to junction trees. In *Conf. on Uncertainty in Artif. Intelligence*, p. 367–373, San Francisco, 1994.

[15] S. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47:1238–1251, 2001.

[16] Ilog Optimization. Cplex documentation. http://www.ilog.com, 1990.

[17] J. D. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *Journal of Artif. Intelligence Research*, 21:101–133, 2004.

[18] R. Qi and D. Poole. A new method for influence diagram evaluation. *Computational Intelligence*, 11:1:1–34, 1995.

[19] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.

[20] N. L. Zhang. Probabilistic inferences in influence diagrams. In *Conf. on Uncertainty in Artif. Intelligence*, p. 514–522, Madison, 1998.

[21] N. L. Zhang and D. Poole. Stepwise-decomposable influence diagram. In *Int. Conf. on Principles of Knowledge Representation and Reasoning*, p. 141–152, Cambridge, 1992.

[22] W. Zhang and Q. Ji. A factorization approach to evaluating simultaneous influence diagrams. *IEEE Transactions on Systems, Man and Cybernetics A*, 36(4):746–757, 2006.

Sensitivity analysis for finite Markov chains in discrete time

Gert de Cooman, Filip Hermans and **Erik Quaeghebeur**
SYSTeMS Research Group
Ghent University, Belgium
{gert.decooman,filip.hermans,erik.quaeghebeur}@UGent.be

Abstract

When the initial and transition probabilities of a finite Markov chain in discrete time are not well known, we should perform a sensitivity analysis. This is done by considering as basic uncertainty models the so-called *credal sets* that these probabilities are known or believed to belong to, and by allowing the probabilities to vary over such sets. This leads to the definition of an *imprecise Markov chain*. We show that the time evolution of such a system can be studied very efficiently using so-called *lower* and *upper expectations*. We also study how the inferred credal set about the state at time n evolves as $n \to \infty$: under quite unrestrictive conditions, it converges to a uniquely invariant credal set, regardless of the credal set given for the initial state. This leads to a non-trivial generalisation of the classical Perron–Frobenius Theorem to imprecise Markov chains.

1 Setting the stage

One convenient way to model uncertain dynamical systems is to describe them as Markov chains. These have been studied in great detail, and their properties are well known. However, in many practical situations, it remains a challenge to accurately identify the transition probabilities in the Markov chain: the available information about physical systems is often imprecise and uncertain. Describing a real-life dynamical system as a Markov chain will therefore often involve unwarranted precision, and may lead to conclusions not supported by the available information.

For this reason, it seems quite useful to perform probabilistic robustness studies, or sensitivity analyses, for Markov chains. This is especially relevant in decision-making applications. Many researchers in Markov Chain Decision Making (White and Eldeib, 1994; Harmanec, 2002; Nilim and El Ghaoui, 2005; Itoh and Nakamura, 2007)—inspired by Satia and Lave's (1973) original work—have paid attention to this issue of 'imprecision' in Markov chains.

Early work on the more mathematical aspects of modelling such 'imprecision' in Markov chains was done by Kozine and Utkin (2002). Armed with linear programming techniques, these authors also performed an experimental study of the limit behaviour of Markov chains with uncertain transition probabilities. More recently, Škulj (2006, 2007) has begun a formal study of the time evolution and limit behaviour of such systems.

All these approaches use *sets of probabilities* to deal with the imprecision in the transition probabilities. When these probabilities are not well known, they are assumed to belong to certain sets, and robustness analyses are performed by allowing the transition probabilities to vary over such sets. As we shall see, this approach leads to a number of computational difficulties, which we show can be overcome by tackling the same problem from another angle, using lower and upper expectations, rather than sets of probabilities.

In the rest of this Introduction, we give an overview of the theory of classical Markov chains and formulate the classical Perron–Frobenius theorem. Then, in Sections 2 and 3, we introduce imprecise Markov chains and generalise many aspects of the classical theory. In Section 4, we generalize the Perron–Frobenius theorem. We discuss a number of theoretical and numerical examples in Section 5, and we give perspectives for further research in the Conclusions.

1.1 Analysis of classical Markov chains

Consider a finite Markov chain in discrete time, where at times $n = 1,2,3,\ldots,N$, $N \in \mathbb{N}$ the *state* $X(n)$ of a system can assume any value in a finite set $\mathscr{X}$. Here $\mathbb{N}$ denotes the set of non-zero natural numbers, and N is the time horizon. The time evolution of such a system can be modelled as if it traversed a so-called *event tree* (Shafer, 1996). An example of such a tree for $\mathscr{X} = \{a,b\}$ and $N = 3$ is given in Figure 1. The *situations*, or nodes, of the tree have the form $x_{1:k} := (x_1,\ldots,x_k) \in \mathscr{X}^k$, $k = 0,1,\ldots,N$. For $k = 0$ there is some abuse of notation as we let $\mathscr{X}^0 := \{\Box\}$, where $\Box$ is the

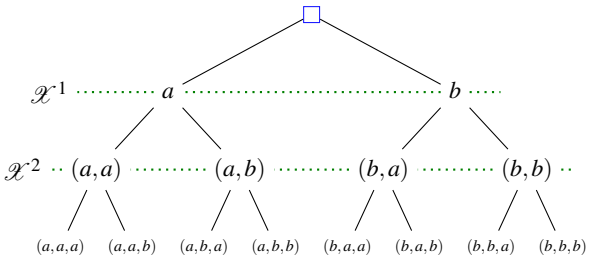


Figure 1: The event tree for the time evolution of system that can be in two states, a and b, and can change state at time instants $n = 1,2$. Also depicted are the respective cuts $\mathscr{X}^1$ and $\mathscr{X}^2$ of $\square$ where the states at times 1 and 2 are revealed.

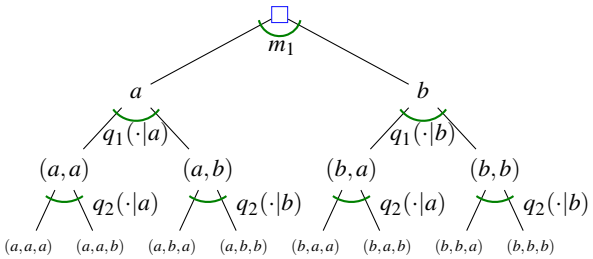


Figure 2: The probability tree for the time evolution of a Markov chain that can be in two states, a and b, and can change state at each time instant $n = 1,2$.

so-called *initial situation*, or root of the tree. In the *cuts* $\mathscr{X}^n$ of $\square$, the value of the state $X(n)$ at time n is revealed.

In a classical analysis, it is generally assumed that we have: (i) a probability distribution over the initial state $X(1)$, in the form of a probability mass function m_1 on $\mathscr{X}$; and (ii) for each situation $x_{1:n}$ that the system can be in at time n, a probability distribution over the next state $X(n+1)$, in the form of a probability mass function $q(\cdot|x_{1:n})$ on $\mathscr{X}$. This means that in each non-terminal situation[1] $x_{1:n}$ of the event tree, we have a *local* probability model telling us about the probabilities of each of its child nodes. This turns the event tree into a so-called *probability tree*; see Shafer (1996, Chapter 3) and Kemeny and Snell (1976, Section 1.9).

The probability tree for a Markov chain is special, because the *Markov Condition* states that when the system jumps from state $X(n) = x_n$ to a new state $X(n+1)$, where the system goes to will only depend on the state $X(n) = x_n$ the system was in at time n, and not on its states $X(k) = x_k$ at previous times $k = 1,2,\ldots,n-1$. In other words:

$$q(\cdot|x_{1:n}) = q_n(\cdot|x_n), x_{1:n} \in \mathscr{X}^n, n = 1,\ldots,N-1, \quad (1)$$

where $q_n(\cdot|x_n)$ is some probability mass function on $\mathscr{X}$. The Markov chain may be non-stationary, as the transition probabilities are allowed to depend explicitly on the time n. Figure 2 gives an example of a probability tree for a Markov chain with $\mathscr{X} = \{a,b\}$ and $N = 3$.

With the local probability mass functions m_1 and $q_n(\cdot|x_n)$ we associate the linear real-valued *expectation functionals* E_1 and $E_n(\cdot|x_n)$, given, for all real-valued maps h on $\mathscr{X}$, by

$$E_1(h) := \sum_{x_1 \in \mathscr{X}} h(x_1) m_1(x_1),$$
$$E_n(h|x_n) := \sum_{x_{n+1} \in \mathscr{X}} h(x_{n+1}) q_n(x_{n+1}|x_n).$$

In any probability tree, probabilities and expectations can be calculated very efficiently using backwards recursion. Suppose that in situation $x_{1:n}$, we want to calculate the conditional expectation $E(f|x_{1:n})$ of some real-valued function f on $\mathscr{X}^N$ that may depend on the values of the states $X(1)$ to $X(N)$. Let us indicate briefly how this is done, also taking into account the simplifications due to the Markov Condition (1). A prominent part is played by the so-called *transition operators* T_n and $\mathbb{T}_n$. Consider the linear space $\mathscr{L}(\mathscr{X})$ of all real-valued maps on $\mathscr{X}$. Then the linear operator $\mathrm{T}_n \colon \mathscr{L}(\mathscr{X}) \to \mathscr{L}(\mathscr{X})$ is defined by

$$\mathrm{T}_n h(x_n) := E_n(h|x_n) = \sum_{x_{n+1} \in \mathscr{X}} h(x_{n+1}) q_n(x_{n+1}|x_n) \quad (2)$$

for all real-valued maps h on $\mathscr{X}$. In other words, $\mathrm{T}_n h$ is the real-valued map on $\mathscr{X}$ whose value $\mathrm{T}_n h(x_n)$ in $x_n \in \mathscr{X}$ is the conditional expectation of the random variable $h(X(n+1))$, given that the system is in state x_n at time n. More generally, we also consider the linear maps $\mathbb{T}_n$ from $\mathscr{L}(\mathscr{X}^{n+1})$ to $\mathscr{L}(\mathscr{X}^n)$, defined by

$$\begin{aligned}\mathbb{T}_n f(x_{1:n}) :=& \mathrm{T}_n f(x_{1:n},\cdot)(x_n) = E_n(f(x_{1:n},\cdot)|x_n) \\ =& \sum_{x_{n+1} \in \mathscr{X}} f(x_{1:n},x_{n+1}) q_n(x_{n+1}|x_n) \end{aligned} \quad (3)$$

for all $x_{1:n}$ in $\mathscr{X}^n$ and all real-valued maps f on $\mathscr{X}^{n+1}$. We begin with the expectation $E(f|x_{1:n})$ for $n = N-1$:

$$\begin{aligned} E(f|x_{1:N-1}) =& \sum_{x_N \in \mathscr{X}} f(x_{1:N-1},x_N) q(x_N|x_{1:N-1}) \\ =& \sum_{x_N \in \mathscr{X}} f(x_{1:N-1},x_N) q_{N-1}(x_N|x_{N-1}) \\ =& \mathbb{T}_{N-1} f(x_{1:N-1}), \end{aligned}$$

where the second inequality follows from the Markov Condition (1), and the third from Eq. (3). Similar arguments for $n = N-2$ and the Law of Iterated Expectations yield:

$$\begin{aligned} E(f|x_{1:N-2}) &= E(E(f(x_{1:N-2},\cdot,\cdot)|x_{1:N-2},\cdot)|x_{1:N-2}) \\ &= \mathbb{T}_{N-2}\mathbb{T}_{N-1} f(x_{1:N-2}). \end{aligned}$$

Repeating this argument leads to the backwards recursion formulae (for $n = 1,\ldots,N-1$)

$$E(f|x_{1:n}) = \mathbb{T}_n \mathbb{T}_{n+1} \ldots \mathbb{T}_{N-1} f(x_{1:n}) \quad (4)$$
$$E(f) := E(f|\square) = E_1(\mathbb{T}_1 \mathbb{T}_2 \ldots \mathbb{T}_{N-1} f). \quad (5)$$

[1] A *non-terminal* situation is a node of the tree that is not a leaf.

In these formulae, f is any real-valued function on $\mathscr{X}^N$. If we let f be the indicator functions $I_{\{x_{1:N}\}}$ of the singletons $\{x_{1:N}\}$, these formulae allow us for instance to calculate the joint probability mass function $p(x_{1:N}) = E(I_{\{x_{1:N}\}})$ for all the variables $X(1), \ldots, X(N)$. We can also use them to find the conditional mass functions $p(x_{n+1:N}|x_n) = p(x_{n+1:N}|x_{1:n}) \coloneqq E(I_{\{x_{1:N}\}}|x_{1:n})$.

1.2 The Perron–Frobenius Theorem for classical Markov chains

We are especially interested in the case of a *stationary* Markov chain, and in the (marginal) expectation $E_n(h)$ of a real-valued function h (on $\mathscr{X}$) that depends only on the state $X(n)$ at time n. Here, Eq. (5) becomes

$$E_n(h) \coloneqq E_1(\mathrm{T}^{n-1}h), \tag{6}$$

where $\mathrm{T} \coloneqq \mathrm{T}_1 = \mathrm{T}_2 = \cdots = \mathrm{T}_{N-1}$, and where we denote by T^k the k-fold composition of T with itself; in particular, T^0 is the identity operator on $\mathscr{L}(\mathscr{X})$. If we let $h = I_{\{x_n\}}$, this allows us to find the probability mass function $m_n(x_n) = E_n(I_{\{x_n\}})$ for the state $X(n)$. Under some restrictions on the transition operator T, the classical Perron–Frobenius Theorem then tells us that, as n and the time horizon N recede to infinity, this probability mass function converges to some limit, independently of the initial probability mass function m_1; see Kemeny and Snell (1976, Theorem 4.1.6) and Luenberger (1979, Chapter 6). In terms of expectation functionals and transition operators:

Theorem 1 (Classical Perron–Frobenius Theorem, Expectation Form)**.** *Consider a stationary Markov chain with finite state set $\mathscr{X}$ and transition operator* T*. Suppose that* T *is regular, meaning that there is some $k > 0$ such that $\min \mathrm{T}^k I_{\{x_k\}} > 0$ for all x_k in $\mathscr{X}$. Then for every initial expectation operator E_1, the expectation operator $E_n = E_1 \circ \mathrm{T}^{n-1}$ for the state at time n converges point-wise to the same limit expectation operator E_∞: for all $h \in \mathscr{L}(\mathscr{X})$,*

$$\lim_{n\to\infty} E_n(h) = \lim_{n\to\infty} E_1(\mathrm{T}^{n-1}h) = E_\infty(h).$$

Moreover, the limit expectation E_∞ is the only T*-invariant expectation on $\mathscr{L}(\mathscr{X})$, in the sense that $E_\infty = E_\infty \circ \mathrm{T}$.*

2 Towards imprecise Markov chains

The treatment above rests on the assumption that the initial probabilities and the transition probabilities are precisely known. If such is not the case, then it seems necessary to perform some kind of sensitivity analysis, in order to find out to what extent any conclusions we might reach using such a treatment, depend on the actual values of these probabilities.

A very general way of performing a sensitivity analysis for probabilities involves calculations with closed convex sets of probability mass functions, also called *credal sets*, rather than with single probability measures. Let $\Sigma_{\mathscr{X}}$ denote the set of all probability mass functions on $\mathscr{X}$, an $(|\mathscr{X}|-1)$-dimensional unit simplex in the $|\mathscr{X}|$-dimensional linear space $\mathbb{R}^{\mathscr{X}}$, then $\{m \in \Sigma_{\mathscr{X}} \colon (\forall x \in \mathscr{X})(m(x) \leq \frac{1}{2})\}$ is a credal set, but $\{m \in \Sigma_{\mathscr{X}} \colon (\exists x \in \mathscr{X})(m(x) \geq \frac{1}{2})\}$ is not.

There is a growing body of literature on this interesting and fairly new area of *imprecise probabilities*, starting with the publication of Walley's (1991) seminal work. We refer to the literature (Walley, 1991, 1996; Weichselberger, 2001; De Cooman and Miranda, 2007) for more details and discussion.

Specifying a closed convex set $\mathscr{P}$ of probability mass functions p on a finite set $\mathscr{Y}$ is equivalent (Walley, 1991, Section 3.4.1) to specifying its *lower* and *upper expectation* (functionals) $\underline{E}_{\mathscr{P}} \colon \mathscr{L}(\mathscr{Y}) \to \mathbb{R}$ and $\overline{E}_{\mathscr{P}} \colon \mathscr{L}(\mathscr{Y}) \to \mathbb{R}$, defined by

$$\underline{E}_{\mathscr{P}}(g) \coloneqq \min\{E_p(g) \colon p \in \mathscr{P}\}$$
$$\overline{E}_{\mathscr{P}}(g) \coloneqq \max\{E_p(g) \colon p \in \mathscr{P}\}$$

for real-valued maps g on $\mathscr{Y}$, where $E_p(g) = \sum_{y\in\mathscr{Y}} g(y)p(y)$ is the expectation of g associated with the probability mass function p. In a sensitivity analysis, such functionals are quite useful, because they give tight lower and upper bounds on the expectation of any real-valued map. Since the functionals $\underline{E}_{\mathscr{P}}$ and $\overline{E}_{\mathscr{P}}$ are *conjugate* in the sense that $\underline{E}_{\mathscr{P}}(g) = -\overline{E}_{\mathscr{P}}(-g)$ for all real-valued maps g on $\mathscr{Y}$, one is completely determined if the other is known. Below, we concentrate on upper expectations.

What is the upshot of all this for the Markov chain problem we are considering here? First of all, in the initial situation $\Box$, corresponding to time $n = 0$, rather than a single initial probability mass function m_1, we now have a local credal set $\mathscr{M}_1$ of candidate mass functions m_1 for the state $X(1)$ that the system will be in at time $k = 1$. We denote by $\overline{E}_1$ the upper expectation associated with $\mathscr{M}_1$:

$$\overline{E}_1(h) \coloneqq \max\Big\{\sum_{x\in\mathscr{X}} h(x)m_1(x) \colon m_1 \in \mathscr{M}_1\Big\}$$

for all $h \in \mathscr{L}(\mathscr{X})$. Also, in any situation $x_{1:n} \in \mathscr{X}^n$, corresponding to time $n = 1, 2, \ldots, N-1$, instead of a single transition mass function $q_n(\cdot|x_n)$, we now have a local credal set $\mathscr{Q}_n(\cdot|x_n)$ of candidate conditional mass functions $q_n(\cdot|x_n)$ for the state $X(n+1)$ that the system will be in at time $n+1$. We denote by $\overline{E}_n(\cdot|x_n)$ the upper expectation associated with $\mathscr{Q}_n(\cdot|x_n)$, i.e., for all $h \in \mathscr{L}(\mathscr{X})$:

$$\overline{E}_n(h|x_n) \coloneqq \max\Big\{\sum_{x\in\mathscr{X}} h(x)q(x) \colon q \in \mathscr{Q}_n(\cdot|x_n)\Big\}. \tag{7}$$

We call the resulting model an *imprecise Markov chain*. Figure 3 gives an example of an imprecise Markov chain probability tree. A classical, or *precise*, Markov chain is an imprecise one with credal sets that are singletons.

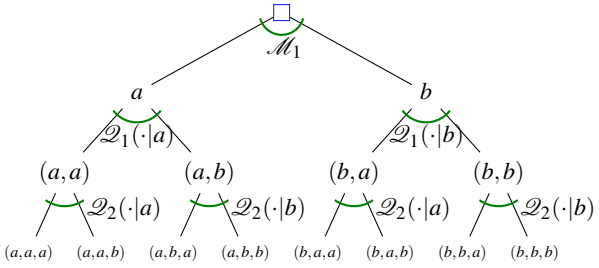


Figure 3: The tree for the time evolution of an imprecise Markov chain that can be in two states, a and b, and can change state at each time instant $n = 1,2$.

How, then, is a sensitivity analysis typically performed (Kozine and Utkin, 2002; Škulj, 2006, 2007) for such an imprecise Markov chain? We choose, in each non-terminal situation $x_{1:k}$ of the above-mentioned event tree, a local transition probability mass $q(\cdot|x_{1:k})$ in the set of possible candidates $\mathscr{Q}_k(\cdot|x_k)$.[2] For $k = 0$, we get the initial situation $\square$, where we choose some element m_1 in the set of possible candidates $\mathscr{M}_1$. We then obtain a so-called *compatible probability tree*, for which we may calculate all (conditional) expectations and probability mass functions:

$$E(f|x_{1:n}) = \sum_{x_{n+1:N}\in\mathscr{X}^{N-n}} f(x_{1:n},x_{n+1:N}) \prod_{k=n}^{N-1} q(x_{k+1}|x_{1:k}), \qquad (8)$$

$$E(f) = \sum_{x_{1:N}\in\mathscr{X}^{N}} f(x_{1:N}) m_1(x_1) \prod_{k=1}^{N-1} q(x_{k+1}|x_{1:k}), \qquad (9)$$

for $n = 1,\dots,N-1$, and for all real-valued maps f on $\mathscr{X}^N$. As we have just come to realise, the probability trees that are compatible with an imprecise Markov chain are no longer necessarily (precise) Markov chains themselves. It is still possible to calculate the $E(f|x_{1:n})$ and $E(f)$ in Eqs. (8) and (9) using backwards recursion (Shafer, 1996, Chapter 3), but the formulae for doing so will be more complicated than the ones for precise Markov chains given by Eqs. (4) and (5).

If we repeat this for every other choice of the m_1 in $\mathscr{M}_1$ and the $q(\cdot|x_{1:k})$ in $\mathscr{Q}_k(\cdot|x_k)$, we end up with an infinity of compatible probability trees, for which the associated (conditional) expectations and probability mass functions turn out to be closed convex sets. We denote their corresponding upper expectation functionals on $\mathscr{L}(\mathscr{X}^N)$ by $\overline{E}(\cdot|x_{1:n})$ and $\overline{E}$. These upper expectations, and the conjugate lower expectations, are the final aim of our sensitivity analysis.

The procedure we have just described is computationally very complex. When the closed convex sets $\mathscr{M}_1$ and $\mathscr{Q}_k(\cdot|x)$ each have a finite number of extreme points (are polytopes), we can limit ourselves to working with these sets of extreme points, rather than with the infinite sets themselves. But even then, the computational complexity of this approach will generally be exponential in the number of time steps.

However, we prove in Section 3 that the upper expectations $\overline{E}$ and $\overline{E}(\cdot|x_{1:n})$ associated with the closed convex sets of (conditional) probability mass functions for the compatible probability trees of an imprecise Markov chain can be calculated in the same way as the expectations E and $E(\cdot|x_{1:n})$ in a precise one: using counterparts of the backwards recursion formulae (4)–(6). Because of this, making inferences about the mass function of the state at time n, i.e., finding the upper envelope $\overline{E}_n$ of the E_n given in Eq. (6) *now has a complexity that is linear, rather than exponential, in the number of time steps n.* This is our first contribution.

Our second contribution in this paper is a Perron–Frobenius Theorem for a special class of so-called regular stationary imprecise Markov chains: in Section 4 we prove a generalisation of Theorem 1, which tells us that under the fairly weak condition of regularity, the upper expectation operators $\overline{E}_n$ converge to limits that do not depend on the initial upper expectation operators $\overline{E}_1$.

3 Sensitivity analysis of imprecise Markov chains

We are now ready to take our most important step: the backwards recursion formulae for the conditional and joint upper expectations in an imprecise Markov chain. We first define *upper transition operators* $\overline{\mathrm{T}}_n$ and $\overline{\mathbb{T}}_n$. The operator $\overline{\mathrm{T}}_n\colon \mathscr{L}(\mathscr{X}) \to \mathscr{L}(\mathscr{X})$ is defined by

$$\overline{\mathrm{T}}_n h(x_n) := \overline{E}_n(h|x_n) \qquad (10)$$

for all real-valued maps h on $\mathscr{X}$, and all x_n in $\mathscr{X}$. In other words, $\overline{\mathrm{T}}_n h$ is the real-valued map on $\mathscr{X}$, whose value $\overline{\mathrm{T}}_n h(x_n)$ in $x_n \in \mathscr{X}$ is the conditional upper expectation of the random variable $h(X(n+1))$, given that the system is in state x_n at time n. More generally, we also consider the maps $\overline{\mathbb{T}}_n$ from $\mathscr{L}(\mathscr{X}^{n+1})$ to $\mathscr{L}(\mathscr{X}^n)$, defined by

$$\overline{\mathbb{T}}_n f(x_{1:n}) := \overline{\mathrm{T}}_n f(x_{1:n},\cdot)(x_n) = \overline{E}_n(f(x_{1:n},\cdot)|x_n) \qquad (11)$$

for all $x_{1:n}$ in $\mathscr{X}^n$ and all real-valued maps f on $\mathscr{X}^{n+1}$. Of course, we can also consider lower expectations and lower transition operators, which are related to the upper expectations and upper transition operators by conjugacy.

The upper expectations $\overline{E}(\cdot|x_{1:n})$ and $\overline{E}$ on $\mathscr{L}(\mathscr{X}^N)$ can be calculated very easily by backwards recursion.

Theorem 2 (Concatenation Formula)**.** *For any $x_{1:n}$ in $\mathscr{X}^n$, $n = 1,\dots,N-1$, and for any real-valued map f on $\mathscr{X}^N$:*

$$\overline{E}(f|x_{1:n}) = \overline{\mathbb{T}}_n\overline{\mathbb{T}}_{n+1}\dots\overline{\mathbb{T}}_{N-1} f(x_{1:n}) \qquad (12)$$

$$\overline{E}(f) = \overline{E}_1(\overline{\mathbb{T}}_1\overline{\mathbb{T}}_2\dots\overline{\mathbb{T}}_{N-1} f). \qquad (13)$$

[2] These local transition probability masses themselves depend on the situation $x_{1:k}$ they are attached to, but the sets $\mathscr{Q}_k(\cdot|x_k)$ they are chosen from only depend on the last state x_k.

Call, for any non-empty subset I of $\{1\ldots,N\}$, a real-valued map f on $\mathscr{X}^N$ *I-measurable* if $f(x_{1:N}) = f(z_{1:N})$ for all $x_{1:N}$ and $z_{1:N}$ in $\mathscr{X}^N$ such that $x_k = z_k$ for all $k \in I$. In other words, an I-measurable f only depends on the states $X(k)$ at times $k \in I$. As an example, an $\{n\}$-measurable map h only depends on the state $X(n)$ at time n, and we identify it with a map on $\mathscr{X}$ (but remember that it acts on states at time n). The following proposition tells us that all upper conditional expectations satisfy a Markov condition.

Proposition 3 (Markov Condition)**.** *Consider an imprecise Markov chain with finite state set $\mathscr{X}$ and time horizon N. Fix $n \in \{1,\ldots,N-1\}$. Let $x_{1:n-1}$ and $z_{1:n-1}$ be arbitrary elements of $\mathscr{X}^{n-1}$, and let $x_n \in \mathscr{X}$. Let f be any $\{n,n+1,\ldots,N\}$-measurable real-valued map on $\mathscr{X}^N$. Then it holds that $\overline{E}(f|x_{1:n-1},x_n) = \overline{E}(f|z_{1:n-1},x_n)$, and therefore we may write $\overline{E}(f|x_{1:n-1},x_n) = \overline{E}_{|n}(f|x_n)$.*

The index '$|n$' is meant to make clear that we are considering an expectation conditional on $X(n) = x_n$.

If we apply the joint upper expectation $\overline{E}$ to maps h that only depend on the state $X(n)$ at time n, we get the *marginal upper expectation* $\overline{E}_n(h) := \overline{E}(h)$, which is a model for the uncertainty about the state $X(n)$ at time n. More generally, taking into account Proposition 3, we use the notation $\overline{E}_{n|\ell}(h|x_\ell) := \overline{E}_{|\ell}(h|x_\ell)$ for the upper expectation of $h(X(n))$, conditional on $X(\ell) = x_\ell$ with $1 \leq \ell < n$. With notations established in Eq. (7), $\overline{E}_{n+1|n}(h|x_n) = \overline{E}_n(h|x_n) = \overline{\mathrm{T}}_n h(x_n)$. Such expectations can be found using simpler recursion formulae than Eqs. (12) and (13), as they are based on the simpler upper transition operators $\overline{\mathrm{T}}_k$.

Corollary 4. *For any real-valued map h on $\mathscr{X}$, and for any $1 \leq \ell < n \leq N$ and all x_ℓ in $\mathscr{X}$:*

$$\begin{aligned}\overline{E}_{n|\ell}(h|x_\ell) &= \overline{\mathrm{T}}_\ell \overline{\mathrm{T}}_{\ell+1} \ldots \overline{\mathrm{T}}_{n-1} h(x_\ell),\\ \overline{E}_n(h) &= \overline{E}_1(\overline{\mathrm{T}}_1 \overline{\mathrm{T}}_2 \ldots \overline{\mathrm{T}}_{n-1} h). \end{aligned} \tag{14}$$

This offers a reason for formulating our theory in terms of real-valued maps rather than events: suppose we want to calculate the upper probability $\overline{E}_n(A)$ that the state $X(n)$ at time n belongs to the set A. According to Eq. (14), $\overline{E}_n(A) = \overline{E}_1(\overline{\mathrm{T}}_1 \ldots \overline{\mathrm{T}}_{n-1} I_A)$, and even if $\overline{\mathrm{T}}_{n-1}$ can still be calculated using upper probabilities only, it will generally assume values other than 0 and 1, and therefore will not be the indicator of some event. Already after one step, i.e., in order to calculate $\overline{\mathrm{T}}_{n-2}\overline{\mathrm{T}}_{n-1} I_A$, we need to leave the ambit of events, and turn to the more general real-valued maps; even if we only want to calculate upper *probabilities* after n steps. But for joint upper and lower probability mass functions, however, we can remain within the ambit of events:

Proposition 5 (Chapman–Kolmogorov Equations)**.** *For an imprecise Markov chain, we have for all $1 \leq n < m \leq N$ and all $(x_n, x_{n+1:m}) \in \mathscr{X}^{m-n+1}$ that*

$$\overline{E}_{|n}(\{x_{n+1:m}\}|x_n) = \prod_{k=n}^{m-1} \overline{\mathrm{T}}_k I_{\{x_{k+1}\}}(x_k), \tag{15}$$

and for all $1 \leq m \leq N$ and all $x_{1:m} \in \mathscr{X}^m$ that

$$\overline{E}(\{x_{1:m}\}) = \overline{E}_1(\{x_1\}) \prod_{k=1}^{m-1} \overline{\mathrm{T}}_k I_{\{x_{k+1}\}}(x_k). \tag{16}$$

Analogous expressions hold for the lower expectations.

4 Convergence for imprecise Markov chains

Let us now consider a *stationary* imprecise Markov chain with infinite horizon, meaning that $\overline{\mathrm{T}}_1 = \overline{\mathrm{T}}_2 = \cdots = \overline{\mathrm{T}}_n = \ldots =: \overline{\mathrm{T}}$. Analogous to the precise case, we define the regularity condition for the upper transition operator $\overline{\mathrm{T}}$. It will turn out to be a sufficient condition for convergence.

Definition 6 (Regularity for upper transition operators)**.** *We call an upper transition operator* $\overline{\mathrm{T}}$ regular *if there is some $n \in \mathbb{N}$ such that $\min \overline{\mathrm{T}}^n I_{\{y\}} > 0$ for all y in $\mathscr{X}$.*

We call an upper expectation $\overline{E}$ on $\mathscr{L}(\mathscr{X})$ $\overline{\mathrm{T}}$-*invariant* whenever $\overline{E} \circ \overline{\mathrm{T}} = \overline{E}$, i.e., if $\overline{E}(\overline{\mathrm{T}} h) = \overline{E}(h)$ for all $h \in \mathscr{L}(\mathscr{X})$. With this definition, we can formulate the upper expectation form of the Perron–Frobenius theorem.

Theorem 7 (Perron–Frobenius Theorem, Upper Expectation Form)**.** *Consider a stationary imprecise Markov chain with finite state set $\mathscr{X}$ and an upper transition operator $\overline{\mathrm{T}}$ that is regular. Then for every initial upper expectation $\overline{E}_1$, the upper expectation $\overline{E}_n = \overline{E}_1 \circ \overline{\mathrm{T}}^{n-1}$ for the state at time n converges point-wise to the same upper expectation $\overline{E}_\infty$:*

$$\lim_{n\to\infty} \overline{E}_n(h) = \lim_{n\to\infty} \overline{E}_1(\overline{\mathrm{T}}^{n-1} h) =: \overline{E}_\infty(h)$$

for all h in $\mathscr{L}(\mathscr{X})$. Moreover, the limit upper expectation $\overline{E}_\infty$ is the only $\overline{\mathrm{T}}$-invariant upper expectation on $\mathscr{L}(\mathscr{X})$.

The classical Perron–Frobenius Theorem (Theorem 1) is of course a special case of our Theorem 7. Škulj (2007) uses a different, credal set approach to prove a similar (but much weaker) result for imprecise Markov chains with regular *lower* transition operators $\underline{\mathrm{T}}$. He also proves a convergence result for conservative (too large) approximations of the $\overline{E}_n$, in the special case that $\overline{\mathrm{T}}$ is regular but 2-alternating; see Section 5.3 for further details.

5 Examples

In this section, we indicate how the above theory can be applied in a number of practical situations, where the upper expectations are of some special types, described in the literature on imprecise probabilities. We present concrete and explicit examples, as well as a number of simulations.

5.1 Contamination models

Suppose we consider a precise stationary Markov chain, with transition operator T. We contaminate it with a vacuous model, i.e., we take a convex mixture with the upper

transition operator $I_{\mathscr{X}}$ max. This leads to the upper transition operator $\overline{\mathrm{T}}$, defined by

$$\overline{\mathrm{T}}h = (1-\varepsilon)\mathrm{T}h + I_{\mathscr{X}}\varepsilon \max h, \tag{17}$$

for all $h \in \mathscr{L}(\mathscr{X})$, where ε is some constant in the open real interval $(0,1)$. The underlying idea is that we consider a specific convex neighbourhood of T. Since for all x in $\mathscr{X}$, $\min \overline{\mathrm{T}} I_{\{x\}} = (1-\varepsilon)\min \mathrm{T} I_{\{x\}} + \varepsilon > 0$, this upper transition operator is always regular, regardless of whether T is! We infer from Theorem 7 that, whatever the initial upper expectation operator $\overline{E}_1$ is, the upper expectation operator $\overline{E}_n$ for the state $X(n)$ will always converge to the same $\overline{E}_\infty$.

What is this $\overline{E}_\infty$ is for given T and ε? For any $n \geq 1$, $\overline{\mathrm{T}}^n h = (1-\varepsilon)^n \mathrm{T}^n h + I_{\mathscr{X}}\varepsilon \sum_{k=0}^{n-1}(1-\varepsilon)^k \max \mathrm{T}^k h$, and therefore

$$\overline{E}_{n+1}(h) = (1-\varepsilon)^n \overline{E}_1(\mathrm{T}^n h) + \varepsilon \sum_{k=0}^{n-1}(1-\varepsilon)^k \max \mathrm{T}^k h. \tag{18}$$

If we now let $n \to \infty$, we see that the limit is indeed independent of the initial upper expectation $\overline{E}_1$:

$$\overline{E}_\infty(h) = \varepsilon \sum_{k=0}^{\infty}(1-\varepsilon)^k \max \mathrm{T}^k h. \tag{19}$$

Example 5.1 (Contaminating a cycle)**.** Consider for instance $\mathscr{X} = \{a,b\}$, and let the precise Markov chain be the cycle with period 2, with transition operator T given by $\mathrm{T}h(a) = h(b)$ and $\mathrm{T}h(b) = h(a)$. Then $\mathrm{T}^{2n}h = h$ and $\mathrm{T}^{2n+1}h = \mathrm{T}h$, and therefore $\max \mathrm{T}^{2n}h = \max \mathrm{T}^{2n+1}h = \max h$, whence $\overline{E}_\infty(h) = \max h$. ♦

Example 5.2 (Contaminating a random walk)**.** Consider a random walk, where $\mathscr{X} = \{a,b\}$ and $\mathrm{T}h = I_{\mathscr{X}} \frac{h(a)+h(b)}{2}$. Then we find that

$$\overline{E}_\infty(h) = \varepsilon \max h + (1-\varepsilon)\frac{h(a)+h(b)}{2}. \qquad ♦$$

Example 5.3 (Another contamination model)**.** To illustrate the convergence properties of an imprecise Markov chain, let us look at a simple numerical example. Again consider $\mathscr{X} = \{a,b\}$ and let the stationary imprecise Markov chain be defined by an initial credal set $\mathscr{M}_1 = \{m \in \Sigma_{\{a,b\}} : 0.6 \leq m(a) \leq 0.9\}$, and a contamination model of the type (17), with $\varepsilon = 0.1$, and for which the precise transition operator T is defined by the Markov matrix $T := \left[\begin{smallmatrix} q(a|a) & q(b|a) \\ q(a|b) & q(b|b) \end{smallmatrix}\right] = \left[\begin{smallmatrix} 0.15 & 0.85 \\ 0.85 & 0.15 \end{smallmatrix}\right]$. In Figure 4 we plot the evolution of $\overline{E}_n(\{a\})$ and $\underline{E}_n(\{a\})$, the upper and lower probability for finding the system in state a at time n, which can be calculated efficiently using Eq. (18).

For comparison, we also plot the evolution of $E_n(\{a\})$, the probability for finding the system in state a at time n, for a (precise) Markov chain defined by probability mass functions that lie on the boundaries of the credal sets defining the above imprecise Markov chain; to wit, its initial mass function is given by $M_1 := \begin{bmatrix} m_1(a) & m_1(b) \end{bmatrix} = \begin{bmatrix} 0.9 & 0.1 \end{bmatrix}$ and its Markov matrix is $\left[\begin{smallmatrix} 0.135 & 0.865 \\ 0.865 & 0.135 \end{smallmatrix}\right]$. Here $E_\infty(\{a\}) = E_\infty(\{b\}) = 0.5$. ♦

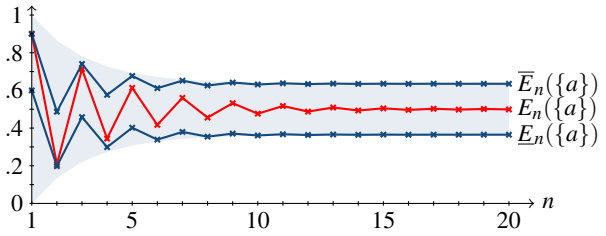


Figure 4: The time evolution of (i) the upper and lower probability of finding the imprecise Markov chain of Example 5.3 in the state a (outer plot marks and connecting lines); and of (ii) the probability of finding the classical Markov chain of Example 5.3 in the state a (inner plot marks and connecting lines). The filled area denotes the hull of the evolution of this probability, under the contamination model of Example 5.3, for all possible initial mass functions.

5.2 Belief function models

The contamination models we have just described are a special case of a more general and quite interesting class of models, based on Shafer's notion of a belief function Shafer (1976). We can consider a number of subsets F_j, $j = 1,\ldots,n$ of $\mathscr{X}$, and a convex mixture of the vacuous upper expectations relative to these subsets:

$$\overline{E}(h) = \sum_{j=1}^{n} m(F_j) \max_{x \in F_j} h(x), \tag{20}$$

with $m(F_j) \geq 0$ and $\sum_{j=1}^{n} m(F_j) = 1$. In Shafer's terminology, the sets F_j are called *focal elements*, and the $m(F_j)$'s the *basic probability assignment*.

We can now consider imprecise Markov chains where the local models, attached to the non-terminal situations in the tree, are of this type. The general backwards recursion formulae we have given in Section 3 can then be used in combination with the simple formulae of the type (20) for an efficient calculation of all conditional and joint upper and lower expectations in the tree. We leave this implicit however, and move on to another example, which is rather more popular in the literature.

5.3 Models with lower and upper mass functions

An intuitive way to introduce imprecise Markov chains (Kozine and Utkin, 2002; Campos et al., 2003; Škulj, 2006) goes by way of so-called *probability intervals*, studied in a paper by De Campos et al. (1994); see also Walley (1991, Section 4.6.1). It consists in specifying lower and upper bounds for mass functions. Let us explain how this is done in the specific context of Markov chains.

For the initial mass function m_1, we specify a lower bound $\underline{m}_1 : \mathscr{X} \to \mathbb{R}$, also called a *lower mass function*, and an upper bound $\overline{m}_1 : \mathscr{X} \to \mathbb{R}$, called an *upper mass function*.

The credal set $\mathscr{M}_1$ attached to the initial situation, which corresponds to these bounds, is then given by

$$\mathscr{M}_1 := \{m \in \Sigma_{\mathscr{X}} : (\forall x \in \mathscr{X})(\underline{m}_1(x) \leq m(x) \leq \overline{m}_1(x))\}.$$

Similarly, in each non-terminal situation $x_{1:k} \in \mathscr{X}^k$, $k = 1, \ldots, N-1$ we have a credal set $\mathscr{Q}_k(\cdot|x_k)$ that is defined in terms of conditional lower and upper mass functions $\underline{q}_k(\cdot|x_k)$ and $\overline{q}_k(\cdot|x_k)$. Here, for instance, $\underline{q}_k(x_{k+1}|x_k)$ gives a lower bound on the transition probability $q_k(x_{k+1}|x_k)$ to go from state $X(k) = x_k$ to state $X(k+1) = x_{k+1}$ at time k.

Under some consistency conditions—see (De Campos et al., 1994) for more details—the upper expectation associated with $\mathscr{M}_1$ is then given in all subsets A of $\mathscr{X}$ by

$$\overline{E}_1(A) = \min\left\{\sum_{z\in A}\overline{m}_1(z), 1 - \sum_{z\in\mathscr{X}\setminus A}\underline{m}_1(z)\right\},$$

This $\overline{E}_1$ is 2-*alternating*: $\overline{E}_1(A\cup B) + \overline{E}_1(A\cap B) \leq \overline{E}_1(A) + \overline{E}_1(B)$ for all subsets A and B of $\mathscr{X}$. This implies (Walley, 1991, Section 3.2.4) that for all $h \in \mathscr{L}(\mathscr{X})$ the upper expectation $\overline{E}_1(h)$ can be found by Choquet integration:

$$\overline{E}_1(h) = \min h + \int_{\min h}^{\max h} \overline{E}_1(\{z \in \mathscr{X} : h(z) \geq \alpha\})\,\mathrm{d}\alpha, \quad (21)$$

where the integral is a Riemann integral. Similar considerations for the 2-alternating $\overline{E}_k(\cdot|x_k)$ lead to formulae for the upper transition operators $\overline{\mathrm{T}}_k$: for all x_k in $\mathscr{X}$,

$$\overline{\mathrm{T}}_k I_A(x_k) = \min\left\{\sum_{z\in A}\overline{q}_k(z|x_k), 1 - \sum_{z\in\mathscr{X}\setminus A}\underline{q}_k(z|x_k)\right\} \quad (22)$$

$$\overline{\mathrm{T}}_k h(x_k) = \min h + \int_{\min h}^{\max h} \overline{\mathrm{T}}_k I_{\{z\in\mathscr{X}\,:\,h(z)\geq\alpha\}}(x_k)\,\mathrm{d}\alpha. \quad (23)$$

Example 5.4 (Close to a cycle)**.** Consider a three-state stationary imprecise Markov model with $\mathscr{X} = \{a,b,c\}$ and with marginal and transition probabilities given by probability intervals. It follows from Eqs. (22) and (23) that the upper transition operator $\overline{\mathrm{T}}$ is fully determined by the upper and lower Markov matrices:

$$\begin{bmatrix} \underline{q}(a|a) & \underline{q}(b|a) & \underline{q}(c|a) \\ \underline{q}(a|b) & \underline{q}(b|b) & \underline{q}(c|b) \\ \underline{q}(a|c) & \underline{q}(b|c) & \underline{q}(c|c) \end{bmatrix} = \frac{1}{200}\begin{bmatrix} 9 & 9 & 162 \\ 144 & 18 & 18 \\ 9 & 162 & 9 \end{bmatrix}$$

$$\begin{bmatrix} \overline{q}(a|a) & \overline{q}(b|a) & \overline{q}(c|a) \\ \overline{q}(a|b) & \overline{q}(b|b) & \overline{q}(c|b) \\ \overline{q}(a|c) & \overline{q}(b|c) & \overline{q}(c|c) \end{bmatrix} = \frac{1}{200}\begin{bmatrix} 19 & 19 & 172 \\ 154 & 28 & 28 \\ 19 & 172 & 19 \end{bmatrix},$$

where the numerical values are particular to this example. Similarly, the initial upper expectation $\overline{E}_1$ is completely determined by the matrices $\overline{M}_1$ and $\underline{M}_1$:

$$\underline{M}_1 := \begin{bmatrix} \underline{m}_1(a) & \underline{m}_1(b) & \underline{m}_1(c) \end{bmatrix}$$
$$\overline{M}_1 := \begin{bmatrix} \overline{m}_1(a) & \overline{m}_1(b) & \overline{m}_1(c) \end{bmatrix}.$$

In Figure 5, we plot conservative approximations for the credal sets $\mathscr{M}_n$ corresponding to the upper expectation operators $\overline{E}_n$. Each approximation is based on the constraints that can be found by calculating $\underline{E}_1(\underline{\mathrm{T}}^{n-1} I_{\{x\}})$ and $\overline{E}_1(\overline{\mathrm{T}}^{n-1} I_{\{x\}})$ using the backwards recursion method, for $x = a,b,c$. The $\mathscr{M}_n$ evolve clockwise through the simplex, which is not all that surprising as the lower and upper Markov matrices are quite 'close' to the precise *cyclic* Markov matrix

$$\begin{bmatrix} q(a|a) & q(b|a) & q(c|a) \\ q(a|b) & q(b|b) & q(c|b) \\ q(a|c) & q(b|c) & q(c|c) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

After a while, the $\mathscr{M}_n$ converge to a limit that is independent of the initial credal set $\mathscr{M}_1$, as can be predicted from the regularity of the upper transition operator. ♦

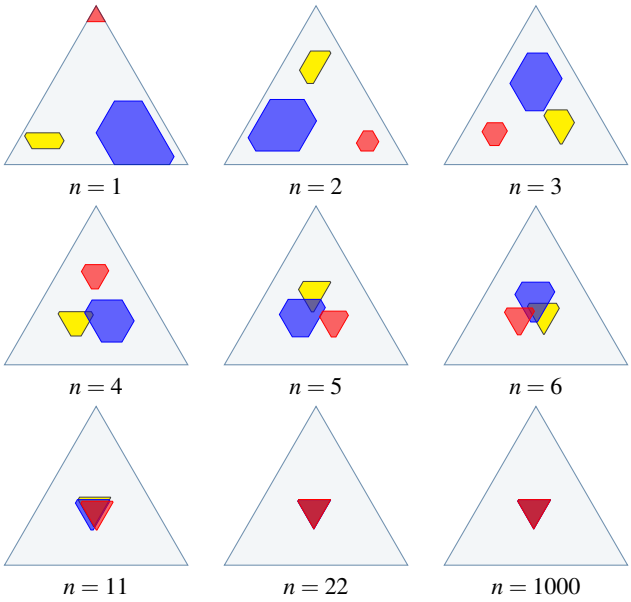


Figure 5: Evolution in the simplex $\Sigma_{\{a,b,c\}}$ of the credal sets $\mathscr{M}_n$ for the near-cyclic transition operator from Example 5.4 for three different choices of the initial credal set $\mathscr{M}_1$.

6 Conclusions

Regularity seems to be a reasonably weak condition on the upper transition operator $\overline{\mathrm{T}}$ for a stationary imprecise Markov chain, but we have seen that it is strong enough to guarantee that the upper expectation for the state at time n converges to a uniquely $\overline{\mathrm{T}}$-invariant upper expectation $\overline{E}_\infty$, regardless of the initial upper expectation $\overline{E}_1$.

Even when the regularity condition is not satisfied, it is not so hard to see that any upper transition operator $\overline{\mathrm{T}}$ is still *non-expansive* under the supremum norm given for every $h \in \mathscr{L}(\mathscr{X})$ by $\|h\|_\infty := \max|h|$:

$$\|\overline{\mathrm{T}}g - \overline{\mathrm{T}}h\|_\infty \leq \|g - h\|_\infty,$$

Moreover, the sequence $\|\overline{\mathrm{T}}^n h\|_\infty$ is bounded because $\|\overline{\mathrm{T}}^n h\|_\infty \leq \|h\|_\infty$. It then follows from non-linear Perron–Frobenius theory (Sine, 1990; Nussbaum et al., 1998) that the sequence $\overline{\mathrm{T}}^n h$ has a periodic limit cycle. More precisely, there is a $\xi_h \in \mathscr{L}(\mathscr{X})$ such that $\overline{\mathrm{T}}^{p_h}\xi_h = \xi_h$ i.e., ξ_h is a *periodic point* of $\overline{\mathrm{T}}$ with (smallest) *period* p_h, and such that $\overline{\mathrm{T}}^{np_h}h \to \xi_h$ (point-wise) as $n \to \infty$. It would be a very interesting topic for further research to study the nature of the periods and periodic points of upper transition operators.

In our discussions, for instance in Section 3, we have consistently used the sensitivity analysis interpretation of imprecise probability models such as upper expectations. Upper and lower expectations can also be given another, so-called *behavioural* interpretation, in terms of some subjects dispositions towards accepting risky transactions. This is for instance Walley's (1991) preferred approach. The results we have derived here remain valid on that alternative interpretation, and the concatenation formulae (12) and (13) can then be shown to be special cases of so-called *marginal extension* procedure (Miranda and De Cooman, 2007), which provides the most conservative coherent (i.e., rational) inferences from the local predictive models $\overline{\mathrm{T}}_k$ to general lower and upper expectations. In another paper (De Cooman and Hermans, 2008), we give more details about how to approach a process theory using imprecise probabilities on a behavioural interpretation.

References

M. A. Campos, G. P. Dimuro, A. C. da Rocha Costa, and V. Kreinovich. Computing 2-step predictions for interval-valued finite stationary Markov chains. Technical Report UTEP-CS-03-20a, University of Texas at El Paso, 2003.

L. M. de Campos, J. F. Huete, and S. Moral. Probability intervals: a tool for uncertain reasoning. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2:167–196, 1994.

Gert de Cooman and Filip Hermans. Imprecise probability trees: Bridging two theories of imprecise probability. *Artificial Intelligence*, 2008. doi: 10.1016/j.artint.2008.03.001. In press.

Gert de Cooman and Enrique Miranda. Symmetry of models versus models of symmetry. In W. L. Harper and G. R. Wheeler, editors, *Probability and Inference: Essays in Honor of Henry E. Kyburg, Jr.*, pages 67–149. King's College Publications, 2007.

D. Harmanec. Generalizing Markov decision processes to imprecise probabilities. *Journal of Statistical Planning and Inference*, 105:199–213, January 2002.

H. Itoh and K. Nakamura. Partially observable Markov decision processes with imprecise parameters. *Artificial Intelligence*, 171:453–490, January 2007.

John G. Kemeny and J. Laurie Snell. *Finite Markov Chains*. Undergraduate Text in Mathematics. Springer-Verlag, New York, 1976. Reprint of the 1960 Edition.

Igor O. Kozine and Lev V. Utkin. Interval-valued finite Markov chains. *Reliable Computing*, 8(2):97–113, April 2002. doi: doi:10.1023/A:1014745904458.

David G. Luenberger. *Introduction to Dynamic Systems. Theory, Models & Applications*. John Wiley & Sons, New York, 1979.

Enrique Miranda and Gert de Cooman. Marginal extension in the theory of coherent lower previsions. *International Journal of Approximate Reasoning*, 46(1):188–225, September 2007. doi: 10.1016/j.ijar.2006.12.009.

A. Nilim and L. El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53:780–798, January 2005.

R. D. Nussbaum, M. Scheutzow, and S. M. Verduyn Lunel. Periodic points of nonexpansive maps and nonlinear generalizations of the Perron–Frobenius theory. *Selecta Mathematica*, 4:141–181, January 1998.

J. K. Satia and R. E. Lave. Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21:728–740, January 1973.

G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, NJ, 1976.

G. Shafer. *The Art of Causal Conjecture*. The MIT Press, Cambridge, MA, 1996.

R Sine. A nonlinear Perron–Frobenius theorem. *Proceedings of the American Mathematical Society*, 109(2):331–336, January 1990.

D. Škulj. Finite discrete time Markov chains with interval probabilities. In J. Lawry, E. Miranda, A. Bugarin, S. Li, M. A. Gil, P. Grzegorzewski, and O. Hryniewicz, editors, *Soft Methods for Integrated Uncertainty Modelling*, pages 299–306. Springer, Berlin, 2006.

D. Škulj. Regular finite Markov chains with interval probabilities. In G. de Cooman, J. Vejnarová, and M. Zaffalon, editors, *ISIPTA '07 – Proceedings of the Fifth International Symposium on Imprecise Probability: Theories and Applications*, pages 405–413. SIPTA, 2007.

P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London, 1991.

P. Walley. Measures of uncertainty in expert systems. *Artificial Intelligence*, 83(1):1–58, May 1996.

K. Weichselberger. *Elementare Grundbegriffe einer allgemeineren Wahrscheinlichkeitsrechnung. I. Intervallwahrscheinlichkeit als umfassendes Konzept*. Physica-Verlag Heidelberg, June 2001.

C. C. White and H. K. Eldeib. Markov decision-processes with imprecise transition-probabilities. *Operations Research*, 42:739–749, January 1994.

Learning Convex Inference of Marginals

Justin Domke
Department of Computer Science
University of Maryland
College Park, MD, 20742
domke@cs.umd.edu

Abstract

Graphical models trained using maximum likelihood are a common tool for probabilistic inference of marginal distributions. However, this approach suffers difficulties when either the inference process or the model is approximate. In this paper, the inference process is first defined to be the minimization of a convex function, inspired by free energy approximations. Learning is then done directly in terms of the *performance* of the inference process at univariate marginal prediction. The main novelty is that this is a direct minimization of empirical risk, where the risk measures the accuracy of predicted marginals.

1 Introduction

This paper concerns the learning and inference of marginal distributions. In inference, some vector $\mathbf{y}$ is observed, and the goal is to approximate univariate marginals $p(x_i|\mathbf{y})$ for some true (unknown) distribution p, and some hidden, discrete vector $\mathbf{x}$. A typical approach to this problem is to fit the parameters of a graphical model to approximate $p(\mathbf{x}|\mathbf{y})$ with some distribution $q(\mathbf{x}|\mathbf{y})$ using the maximum (conditional) likelihood criterion. Then, in inference, a marginalization algorithm is used to compute $q(x_i|\mathbf{y})$.

This approach is often well-justified– given a correct model, $q(\mathbf{x}|\mathbf{y})$ will converge to $p(\mathbf{x}|\mathbf{y})$ in the high data limit. If exact marginalization is possible, the estimates $q(x_i|\mathbf{y})$ will also converge to the true marginals. However, in practice there are often two major problems.

1. **Computational Intractability**. In general, exact inference is intractable in graphical models with high treewidth (such as "grids"), forcing the use of approximate algorithms. For undirected models, where maximum likelihood learning algorithms require repeated inference, learning will also be intractable. (Maximum likelihood learning of a directed model is usually tractable, even when inference is not.) Even if exact learning is feasible, it is unclear if the results are the best, under *approximate* inference.

2. **Model Defects**. Usually in practice, the model is not exactly correct. (That is, the set of conditional independencies asserted by the graph are not exactly true, or the parametrization of individual factors is imperfect.) This means the parameters *cannot* converge to the "true parameters", since the true distribution is not representable. It is known in this case that maximum likelihood learning will converge to representable distribution with minimum KL-divergence to the true distribution. This is different from the distribution that gives the best predicted marginals, even assuming exact inference.

This paper seeks to ameliorate both of the above issues. This is done by, first, fixing the inference step to be a (presumed tractable) minimization of a convex function. The learning step then consists of fitting the parameters of that function such that the *performance of the inference process* at marginal prediction is best. Specifically, this paper suggests learning the parameters of some function $F(\mathbf{y}, \{b_r(\mathbf{x}_r)\})$. For any observation $\mathbf{y}$, F must be convex over the set of local "pseudomarginals", or "beliefs" $\{b_r(\mathbf{x}_r)\}$.

Since F is convex over $\{b_r(\mathbf{x}_r)\}$, it is nothing more than an implicit mapping from observed variables to beliefs. Thought of as a mapping, it is natural to think that the parameters of F could be adjusted to align these predictions with training data. It turns out that this is true, and moreover, the agreement between predictions and true values can be quantified in terms of a user-specified loss function. This is the real advan-

tage of this approach– in learning, the parameters are directly fit to give good marginal predictions, as specified by the loss function. However, a consequence is that only a *mapping* is learned. The approach is not equivalent to approximating the conditional distribution $p(\mathbf{x}|\mathbf{y})$. Nevertheless, predicted marginals are all that are needed for many problems, and experiments suggest this approach can give good results in this case.

2 Inference

This paper is based on minimizing a class of convex functions F motivated by free energy approximations (see Section 4).

$$F = \sum_{f \in \mathcal{F}} \sum_{r \in \mathcal{R}} \sum_{\mathbf{x}_r} w_f(\mathbf{x}_r, \mathbf{y}_r) f(b_r(\mathbf{x}_r)) \tag{1}$$

$\mathcal{F}$ is a set of convex functions of local beliefs. A typical example would be the identity function and the negative entropy function (i.e. $\mathcal{F} = \{b, b \log b\}$). However, in principle, any function that is convex over the interval $(0, 1)$ may be used.

$\mathcal{R}$ is a set of regions. For example, set of of indices of $\mathbf{x}$ in the regions might be the union of the set of cliques $\mathcal{C}$ and the set of individual indices $\mathcal{I}$[1].

The "weighting" functions w_f determine the behavior of F. Fitting the model corresponds to fitting these weighting functions.

Given some observation $\mathbf{y}$, the beliefs are given by minimizing F.

$$\{b_r^*(\mathbf{x}_r)\} = \arg \min_{\{b_r(\mathbf{x}_r)\}} F(\mathbf{y}, \{b_r(\mathbf{x}_r)\}) \tag{2}$$

This function needs to be minimized subject to some constraints. In this paper, the beliefs will be constrained to be "locally consistent". If again the regions are cliques $\mathcal{C}$ and individual indices $\mathcal{I}$, the constraints are:

$$\forall c \in \mathcal{C}, i \in c, x_i, \quad \sum_{\mathbf{x}_{c \setminus i}} b_c(\mathbf{x}_c) = b_i(x_i) \tag{3}$$

$$\forall c \in \mathcal{C}, \quad \sum_{\mathbf{x}_c} b_c(\mathbf{x}_c) = 1 \tag{4}$$

$$\forall i \in \mathcal{I}, \quad \sum_{x_i} b_i(x_i) = 1 \tag{5}$$

$$\forall c \in \mathcal{C}, \mathbf{x}_c, \quad b_c(\mathbf{x}_c) \geq \mathbf{0} \tag{6}$$

$$\forall i \in \mathcal{I}, x_i, \quad b_i(x_i) \geq 0 \tag{7}$$

Since each function in $\mathcal{F}$ is assumed to be convex, F will also be, provided that the weighting functions w_f are positive. (The weights for the linear function $f(b) = b$ need not be constrained). Since the constraints are also convex, this is a convex optimization problem.

It is typical to solve problems like this through the use of "message passing" algorithms. However, in this paper, a more abstract representation will be convenient. Briefly, it is not hard to see that it is possible to transform the problem into the form

$$\mathbf{b}^* = \arg \min_{\mathbf{b}} \sum_{f \in \mathcal{F}} \mathbf{w}_f(\mathbf{y})^T f(\mathbf{b}) \tag{8}$$

$$\text{such that} \quad A\mathbf{b} = \mathbf{d}$$
$$\mathbf{b} \geq \mathbf{0}.$$

Here, the symbol $\mathbf{b}$ is reused (in boldface) to denote a vector containing all beliefs $b_r(\mathbf{x}_r)$ for all regions r and configurations $\mathbf{x}_r$. Similarly, $\mathbf{w}_f(\mathbf{y})$ represents the vector of weights $w_f(\mathbf{x}_r, \mathbf{y}_r)$ for all regions and configurations. To avoid confusion, boldface ($\mathbf{b}$ or $\mathbf{w}_f(\mathbf{y})$) will always be used when referring to the formulation in Eq. 8, while non-boldface ($b_r(\mathbf{x}_r)$ or $w_f(\mathbf{x}_r, \mathbf{y}_r)$) when referring the the original formulation in Eq. 1.

It is easy to see that each of the three linear constraints (Eqs. 3, 4, and 5) can be handled by setting one row of A and one entry of $\mathbf{d}$.

3 Learning

The weighting functions w_f, are assumed to depend on some parameters $\boldsymbol{\theta}$. Thus, learning consists of fitting $\boldsymbol{\theta}$. Now, a loss function needs to be specified to quantify the performance of the predicted marginals. Take a set of samples $\{(\hat{\mathbf{x}}, \hat{\mathbf{y}})\}$ drawn from some true (unknown) distribution $p(\mathbf{x}, \mathbf{y})$. Broadly speaking, we would like that the marginals produced by the minimization of F "tend to match" those of p. Two ways to quantify this are the univariate "log-loss" and "quad-loss".

[1] By a slight abuse of notation, $\mathbf{x}_r$ and $\mathbf{y}_r$ denote "the variables of $\mathbf{x}$ in region r", and "the variables of $\mathbf{y}$ in region r", respectively. So, for example, some region might contain only a single variable of $\mathbf{x}$, but multiple variables of $\mathbf{y}$.

3.1 Log-loss

Suppose that we would like to minimize the sum of expected KL-divergences between the true marginals, and those produced by the convex optimization. Thus, we would like to choose F by minimizing the "risk"

$$F^* = \arg\min_F \sum_{\mathbf{y}} p(\mathbf{y}) \sum_i \sum_{x_i} p(x_i|\mathbf{y}) \log \frac{p(x_i|\mathbf{y})}{b_i^*(x_i|\mathbf{y}, F)}. \tag{9}$$

Here, $b_i^*(x_i|\mathbf{y}, F)$ denotes the belief obtained for variable i by minimizing F on evidence $\mathbf{y}$. Of course, p is unknown, and so the true risk cannot be minimized. Instead, one can attempt a Monte-Carlo estimate of the true risk, and minimize the "empirical risk".

$$F^* = \arg\min_F - \sum_{\mathbf{y}} p(\mathbf{y}) \sum_i \sum_{x_i} p(x_i|\mathbf{y}) \log b_i^*(x_i|\mathbf{y}, F) \tag{10}$$

$$= \arg\min_F - \sum_{\mathbf{y}} \sum_i \sum_{x_i} p(x_i, \mathbf{y}) \log b_i^*(x_i|\mathbf{y}, F) \tag{11}$$

$$\approx \arg\min_F \sum_{\{(\hat{\mathbf{x}}, \hat{\mathbf{y}})\}} \underbrace{-\sum_i \log b_i^*(\hat{x}_i|\hat{\mathbf{y}}, F)}_{L_{\log}} \tag{12}$$

Statistical learning theory studies the conditions under which the approximation represented by Eq. 12 converges in the infinite data limit to that of Eq. 11. However, this paper will not address this.

Now, in general, the minimum in Eq. 12 is found by optimizing with respect to some parameters of F. Consider the derivative of $L_{\log}$ with respect to some parameter θ_i of F.

$$\frac{\partial L_{\log}}{\partial \theta_j} = -\sum_i \frac{\partial b_i^*(\hat{x}_i|\hat{\mathbf{y}}, F)/\partial \theta_j}{b_i^*(\hat{x}_i|\hat{\mathbf{y}}, F)} \tag{13}$$

The issue of how to calculate $\partial b_i^*(\hat{x}_i|\hat{\mathbf{y}}, F)/\partial \theta_j$ will be addressed below.

Kakade et al. (2002) introduced a learning criteria equivalent to $L_{\log}$, and an optimization method for the case of exact inference.

3.2 Quad-loss

An alternative criterion would be to try to minimize the expected squared difference between the true marginals and the estimated marginals.

$$F^* = \arg\min_F \sum_{\mathbf{y}} p(\mathbf{y}) \sum_i \sum_{x_i} (p(x_i|\mathbf{y}) - b_i^*(x_i|\mathbf{y}, F))^2 \tag{14}$$

Again, this true risk can be approximated with an empirical risk.

$$F^* = \arg\min_F \sum_{\mathbf{y}} \sum_i \sum_{x_i} \left(-2p(x_i, \mathbf{y}) b_i^*(x_i|\mathbf{y}, F) + p(\mathbf{y}) b_i^*(x_i|\mathbf{y}, F)^2\right) \tag{15}$$

$$\approx \arg\min_F \sum_{\{(\hat{\mathbf{x}}, \hat{\mathbf{y}})\}} \underbrace{\sum_i \left(-2b_i^*(\hat{x}_i|\hat{\mathbf{y}}, F) + \sum_{x_i} b_i^*(x_i|\hat{\mathbf{y}}, F)^2\right)}_{L_{\text{quad}}} \tag{16}$$

Notice that the second term in Eq. 16 does not depend on the observed data $\{\hat{\mathbf{x}}\}$.

Again, it is not hard to calculate the derivative of L_{quad} with respect to some parameter θ_j of F.

$$\frac{\partial L_{\text{quad}}}{\partial \theta_j} = 2 \sum_i \Big(-\frac{\partial b_i^*(\hat{x}_i|\hat{\mathbf{y}}, F)}{\partial \theta_j} + \sum_{x_i} b_i^*(x_i|\hat{\mathbf{y}}, F) \frac{\partial b_i^*(x_i|\hat{\mathbf{y}}, F)}{\partial \theta_j} \Big) \tag{17}$$

3.3 Derivatives of Beliefs

The above discussion assumes that it is possible to calculate the derivative of the beliefs with respect to the parameters of F. (Recall that the weights $\mathbf{w}_f$ are parameterized by some vector $\boldsymbol{\theta}$.) These derivatives are not obvious, given that the beliefs are determined only implicitly by the minimization of F. To calculate them, it is first necessary to establish two claims.

Claim 1: Let $F(\mathbf{b}, \boldsymbol{\theta})$ be a continuous function such that for all $\boldsymbol{\theta}$, F that has a unique fixed point in $\mathbf{b}$. Define $\mathbf{b}^*(\boldsymbol{\theta})$ such that $\frac{\partial F(\mathbf{b}^*(\boldsymbol{\theta}), \boldsymbol{\theta})}{\partial \mathbf{b}} = \mathbf{0}$. Then,

$$\frac{\partial \mathbf{b}^*(\boldsymbol{\theta})}{\partial \theta_j} = -\Big(\frac{\partial^2 F(\mathbf{b}^*(\boldsymbol{\theta}), \boldsymbol{\theta})}{\partial \mathbf{b} \partial \mathbf{b}^T}\Big)^{-1} \frac{\partial^2 F(\mathbf{b}^*(\boldsymbol{\theta}), \boldsymbol{\theta})}{\partial \mathbf{b} \partial \theta_j}.$$

This claim is essentially a restatement of the (multivariate) Implicit Function Theorem. Here $\frac{\partial^2 F}{\partial \mathbf{b} \partial \mathbf{b}^T}$ denotes the matrix of second partial derivatives of F with respect to the elements of $\mathbf{b}$. Similarly, $\frac{\partial^2 F}{\partial \mathbf{b} \partial \theta_j}$ denotes the vector of partial derivatives of $\partial F/\partial \theta_j$ with respect to the elements of $\mathbf{b}$. Finally, $\partial \mathbf{b}^*/\partial \theta_j$ denotes the vector of derivatives of the elements of $\mathbf{b}^*$, all with respect to θ_j.

This result is not quite adequate to get the derivatives of beliefs, because it does not consider the constraints.

In the following claim, the argument of $(\mathbf{b}^*(\boldsymbol{\theta}), \boldsymbol{\theta})$ to F is dropped for space.

Claim 2: Define $\mathbf{b}^*(\boldsymbol{\theta}) \doteq \arg\min_{\mathbf{b}} F(\mathbf{b}, \boldsymbol{\theta})$, such that $A\mathbf{b} = d$ for some convex function F. Then,

$$\frac{\partial \mathbf{b}^*(\boldsymbol{\theta})}{\partial \theta_j} = (D^{-1}A^T(AD^{-1}A^T)^{-1}AD^{-1} - D^{-1})\frac{\partial^2 F}{\partial \mathbf{b} \partial \theta_j}$$

where $D = (\frac{\partial^2 F}{\partial \mathbf{b} \partial \mathbf{b}^T})$.

A proof is in the appendix. Essentially, the proof consists of augmenting the set of variables with a vector of Lagrange multipliers to enforce that $A\mathbf{b} = \mathbf{d}$, and then applying Claim 1 to the full set of variables $\{\mathbf{b}, \boldsymbol{\lambda}\}$. (See below for the constraint that $\mathbf{b} \geq 0$.)

For the function of interest in this paper, use the formulation of F in Eq. 8.

$$(\frac{\partial^2 F}{\partial \mathbf{b} \partial \mathbf{b}^T}) = D = \text{diag}(\sum_f \mathbf{w}_f(\mathbf{y}) \odot f''(\mathbf{b})) \tag{18}$$

Here, $\odot$ denotes element-wise multiplication, and f'' denotes the second derivative of f. Notice that by virtue of being diagonal, inverting D is trivial and only consists of inverting each entry.

The last term in Claim 2 is also easy to calculate.

$$\frac{\partial F}{\partial \mathbf{b} \partial \theta_j} = \sum_f \frac{\partial \mathbf{w}_f(\mathbf{y})}{\partial \theta_j} \odot f' \tag{19}$$

Finally, the partial derivatives $\partial \mathbf{w}_f(\mathbf{y})/\partial \theta_j$ are determined directly by $\partial w_f(\mathbf{x}_c, \mathbf{y}_c)/\partial \theta_j$. The exact form, of course, depends on the way in which the weighting functions $w_f(\mathbf{x}_c, \mathbf{y}_c)$ are parametrized. In the common case where each of the weighting functions is fully and independently parametrized,

$$\frac{\partial w_f(\mathbf{x}_c, \mathbf{y}_c)}{\partial \theta_{(f', \mathbf{x}'_c, \mathbf{y}'_c)}} = \delta(f = f', \mathbf{x}_c = \mathbf{x}'_c, \mathbf{y}_c = \mathbf{y}'_c),$$

and so the derivatives $\partial \mathbf{w}_f(\mathbf{y})/\partial \theta_j$ are sparse, binary vectors.

The above discussion did not consider the constraint $\mathbf{b} \geq \mathbf{0}$. For the functions $\mathcal{F}$ used in this paper, it is easy to show that if $\mathbf{b}^* = \arg\min_{\mathbf{b}} f(\mathbf{b}, \boldsymbol{\theta})$, then $\mathbf{b}^* > 0$. (This is established by forming the Lagrangian enforcing $A\mathbf{b} = \mathbf{d}$ (Eq. 28) and taking the gradient with respect to $\mathbf{b}$, which must be zero.)

4 A Free Energy Justification

Suppose there is some true distribution over variables $\mathbf{x}$ and $\mathbf{y}$, where each variable is independent of all other variables, given some set of neighbors. Then, by the Hammersley Clifford theorem, the joint distribution can be represented by

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp(\sum_c E(\mathbf{x}_c, \mathbf{y}_c)), \tag{20}$$

where the sum is over the set of all cliques c in the neighborhood graph. It follows that the conditional distribution can be written as a "Conditional Random Field" (Lafferty et al., 2001), i.e.

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} = \frac{1}{Z(\mathbf{y})} \exp(\sum_c E(\mathbf{x}_c, \mathbf{y}_c)), \tag{21}$$

where $Z(\mathbf{y}) = \sum_{\mathbf{x}} \exp(\sum_c E(\mathbf{x}_c, \mathbf{y}_c))$. Notice that if there are any cliques that contain only variables in $\mathbf{y}$, they can be dropped from Eq. 21.

Now, consider inference. Some vector $\mathbf{y}$ is observed, and we are interested in the conditional distribution induced over $\mathbf{x}$. One approach is to minimize the KL-divergence between some distribution $b(\mathbf{x})$ and $p(\mathbf{x}|\mathbf{y})$.

$$\begin{aligned} b^*(\mathbf{x}) &= \arg\min_b \sum_{\mathbf{x}} b(\mathbf{x}) \log \frac{b(\mathbf{x})}{p(\mathbf{x}|\mathbf{y})} && (22) \\ &= \arg\min_b \sum_{\mathbf{x}} b(\mathbf{x}) \log b(\mathbf{x}) && (23) \\ &\quad - \sum_{\mathbf{x}} \sum_c b(\mathbf{x}) E(\mathbf{x}_c, \mathbf{y}_c) + \sum_{\mathbf{x}} b(\mathbf{x}) \log Z(\mathbf{y}) \\ &= \arg\min_b \sum_{\mathbf{x}} b(\mathbf{x}) \log b(\mathbf{x}) && (24) \\ &\quad - \sum_c \sum_{\mathbf{x}_c} b(\mathbf{x}_c) E(\mathbf{x}_c, \mathbf{y}_c) \end{aligned}$$

The second term in Eq. 24, called the average energy, can be computed exactly. However, the first term, known as the entropy, is generally too expensive to compute exactly. In fact, even to represent an arbitrary distribution $b(\mathbf{x})$ will be impossible when the dimension of $\mathbf{x}$ is large. The typical way out of this difficulty is to represent only local marginal distributions, and then to approximate the entropy with a Bethe or Kikuchi approximation (Yedidia et al. 2005). One such approximation would be that

$$\begin{aligned} \sum_{\mathbf{x}} b(\mathbf{x}) \log b(\mathbf{x}) \approx &\sum_{c \in \mathcal{C}} n_c \sum_{\mathbf{x}_c} b(\mathbf{x}_c) \log b(\mathbf{x}_c) \\ &+ \sum_{i \in \mathcal{I}} n_i \sum_{x_i} b(x_i) \log b(x_i), \end{aligned} \tag{25}$$

for appropriate "counting numbers" n_c and n_i chosen for each clique and variable, respectively. Specifically, the Bethe approximation consists of choosing $n_c = 1$,

and n_i is 1 minus the number of cliques in which node i is contained, i.e. $n_i = 1 - |\{c : i \in c\}|$. More sophisticated approximations consider different regions and different counting numbers.

For a singly-connected graph, the Bethe approximation is exact. For general graphs, it is not always clear what choice of counting numbers will result in a good approximation to the entropy or (more importantly) accurate estimated marginals.

The constraints given in Section 2 (Eqs. 3-7) are, in general, a convex relaxation of the "marginal polytope". In general, given a set of local marginal distributions $\{b_r(\mathbf{x}_r)\}$, no global distribution $b(\mathbf{x})$ that satisfies them all (Yedidia et al. 2005). In order to guarantee that a global distribution exists, one must impose a number of linear constraints, yielding a polytope (Wainwright and Jordan 2003). However, for graphs with high treewidth, the number of constraints is very large, which motivates only enforcing a subset of the constraints. (The local consistency constraints are exact in the case of a tree-structured graph.) This can be a problem since the KL-divergence justification given above assumes that the minimization is over *valid* distributions $b(\mathbf{x})$. There will usually exist a set of inconsistent marginals that have a lower estimated KL-divergence than any true distribution, resulting in less accurate predictions.

The above discussion motivates learning the parameters from data. Rather than attempting to approximate the true $E(\mathbf{x}_c, \mathbf{y}_c)$ (if any), and the true entropy, one could fit the parameters to give the best *predicted marginals*, in light of the relaxation to the marginal polytope, as well as any model defects. This would suggest fitting $E(\mathbf{x}_c, \mathbf{y}_c)$, as well as n_c and n_i, where the mapping from the observation $\mathbf{y}$ to the predicted marginals are given by

$$\{b_r^*\} = \arg\min_{\{b_r\}} \sum_c \sum_{\mathbf{x}_c} -b_c(\mathbf{x}_c) E(\mathbf{x}_c, \mathbf{y}_c) + \sum_c n_c \sum_{\mathbf{x}_c} b_c(\mathbf{x}_c) \log b_c(\mathbf{x}_c) + \sum_i n_i \sum_{x_i} b_i(x_i) \log b(x_i). \quad (26)$$

In this paper, this is generalized by, rather than taking a constant counting number (n_c or n_i) for each region, taking a weight that is allowed to depend on $\mathbf{y}_c$ and $\mathbf{x}_c$. This is not suggested by the free energy approximation, but the experiments below suggest it can enable a more accurate approximation of the marginals. Second, rather than just taking terms with b or $b \log b$, consider arbitrary convex functions of the beliefs. This results in the form for F introduced above,

$$\{b_r^*\} = \arg\min_{\{b_r\}} \sum_{f \in \mathcal{F}} \sum_{r \in \mathcal{R}} \sum_{\mathbf{x}_r} w_f(\mathbf{x}_r, \mathbf{y}_r) f(b(\mathbf{x}_r)). \quad (27)$$

Table 1: Test errors with 30% noise. (All per pixel)

Method\Error	Classif.	Regress.	$L_{\log}$	L_{quad}
Pseudo.+M.F.	0.0475	0.0445	0.2705	-0.9109
Pseudo.+B.P.	0.0495	0.0425	0.2109	-0.9149
Mean Field	0.0300	0.0265	0.1265	-0.9470
Belief Prop.	0.0373	0.0314	0.1393	-0.9371
$L_{\log}$	0.0261	0.0201	.0694	-0.9598
L_{quad}	0.0260	0.0201	.0723	-0.9598

Table 2: Test errors with 50% noise. (All per pixel)

Method\Error	Classif.	Regress.	$L_{\log}$	L_{quad}
Pseudo.+M.F.	0.0954	0.0920	0.6291	-0.8161
Pseudo.+B.P.	0.0977	0.0866	0.5056	-0.8268
Mean Field	0.0751	0.0668	0.3030	-0.8663
Belief Prop.	0.0783	0.0671	0.3039	-0.8658
$L_{\log}$	0.0575	0.0417	0.1384	-0.9165
L_{quad}	0.0561	0.0414	0.1428	-0.9172

This is also less general than normal free energy approximations, in the sense that the weights for the $f = b \log b$ term are constrained to be positive, while counting numbers can be negative.

5 Experiments

As a basic test of this approach, the algorithm was used to learn to "denoise" binary images of handwritten digits from the MNIST database. Ten images of the digits 1-9 were randomly selected for training and testing datasets. Each of the images was then subjected to various amounts of noise. For example, 10% noise means that each pixel has a 10% chance of being assigned randomly. The noisy images make up the observed vectors $\mathbf{y}$, while the original images make up the hidden vectors $\mathbf{x}$.

The model uses regions consisting of individual nodes of $\mathbf{x}$ and $\mathbf{y}$ at the same location, and pairs of neighboring nodes, also at the same locations. The same weights were used for horizontal and vertical regions. The weights were not constrained to be symmetric.

The PDCO primal-dual interior method[2] was used to optimize F. Since the functions f are twice differentiable, the full (diagonal) Hessian can be calculated in

[2] www.stanford.edu/group/SOL/software/pdco.html

Table 3: $w_{b \log b}(\mathbf{x}_c, \mathbf{y}_c)$ and $w_{b \log b}(x_i, y_i)$ for L_{quad} with 50% noise

$\mathbf{x}_c \backslash \mathbf{y}_c$	(0,0)	(0,1)	(1,0)	(1,1)
(0,0)	4.86	0.04	0.05	0.02
(0,1)	4.22	3.86	4.54	5.00
(1,0)	4.13	4.49	2.14	5.13
(1,1)	0.06	0.02	0.03	0.02

$x_i \backslash y_i$	0	1
0	4.47	0.02
1	0.03	0.03

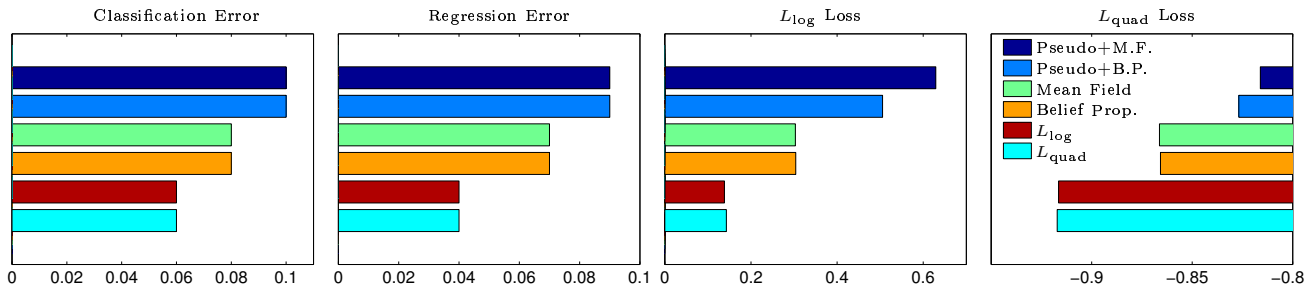


Figure 1: Various measures of accuracy for the six compared methods, with 50% noise.

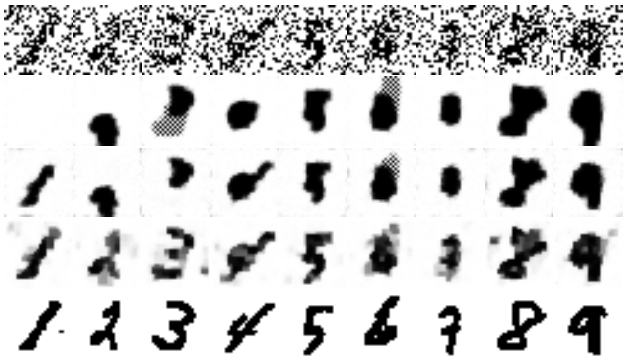


Figure 2: Example results with 50% noise. Top row: Randomly chosen noisy input images of the digits 1-9 from the test set. Second Row: Results from Pseudo.+B.P. Third row: Results from Belief Prop. Fourth Row: Beliefs from convex inference with $L_{\log}$ learning. Bottom row: Original images without noise.

closed form.

The method prosed here was used, with both loss functions $L_{\log}$, and L_{quad}, using the set of functions $\mathcal{F} = \{b, b\log b\}$. The weighting functions are fully parametrized.

Using the results of Section 3 and the "inner-loop" optimization of F above, it is possible to compute empirical risk $\sum_{\{(\hat{\mathbf{x}},\hat{\mathbf{y}})\}} L$ and its gradient $\sum_{\{(\hat{\mathbf{x}},\hat{\mathbf{y}})\}} \partial L/\partial \boldsymbol{\theta}$. An "outer-loop" optimization can use these computation to minimize the empirical risk. Experimentally, small numerical fluctuations in the inner loop can cause instability in the overall optimization. To reduce this, the tolerances on the optimization over F were set very conservatively. Then, a quasi-Newton method (BFGS) was used for the outer optimization.

All weights for the b and $b\log b$ terms are initialized to 0 and 1, respectively. First, 100 iterations are taken, with the weights $w_{b\log b}$ frozen to one, and then another 100 iterations are taken for all weights. This heuristic seems to somewhat improve convergence.

These models are compared against the CRF implementation[3] of Vishwanthan et al. (2006). The features for individual nodes were a constant, and binary variables corresponding to all the possible values of y_i at the same location. In the notation of that paper, $\mathbf{h}_i = [1, \delta(y_i = -1), \delta(y_i = +1)]$. The edge features are similar– a constant, as well as binary features corresponding to each of the four possible configurations of the noisy pixels at the same locations. Four CRF approaches were considered: Pseudolikelihood learning with mean field or belief propagation inference, and mean-field and belief-propagation[4], using those algorithms to approximate marginals for learning and then again for inference. The BFGS method was used for optimization. If the inference method failed to converge after 10,000 iterations, it was stopped at that point.

The approaches are evaluated with four different metrics: the losses $L_{\log}$ and L_{quad}, the "classification error", and the "regression error". Classification error measures the performance in the case that a single prediction (0 or 1) needs to be made for each pixel. (This is done by choosing the value with the highest computed marginal probability.) The classification error is the fraction of predictions that are incorrect. Regression error measures the performance of the model at predicting expected values. The error is the sum of squared differences $(\hat{x}_i - b^*(x_i = 1|\hat{\mathbf{y}}))^2$. Notice that this is different from L_{quad}, as L_{quad} tries to measure the squared difference of *marginal probabilities.*

Tables 1 and 2 give the test losses for 30% and 50% noise. Figure 1 give a graphical representation of the errors for 50% noise. Here, learning to minimize the losses $L_{\log}$ and L_{quad} in general perform quite similarly (though each does a slightly better job of minimizing its own loss.) In general, these significantly outperform learning and inference with belief propagation, which in turn outperform pseudolikelihood based learning. Figure 2 gives example outputs where the output be-

[3] www.cs.ubc.ca/~murphyk/Software/CRF/crf.html

[4] In all cases, "belief propagation" refers to the sum-product formulation.

liefs are visualized with a greyscale intensity.

An interesting question is if allowing the weighting function for the entropy term to vary over the configurations (as opposed to a constant counting number) improves predictions. Table 3 shows the values found for 50% noise. In this particular case at least, the values are highly variant over the configurations, suggesting that this flexibility *is* helpful.

6 Related Work

The basic idea that when approximate inference must be used, the learning process should be cognizant of this has been suggested several times. Domingos (2007) suggested "Deep Combination of Learning and Inference" as one of the most important problems problems of the next ten years.

Specific related work demonstrating this principle includes Wainwright (2006) on the value of an inconsistent estimator in the context of approximate inference. This paper creates a tractable surrogate to the entropy. This same surrogate is used for both learning and inference in such a way that the errors of the two processes can cancel each other to some degree – the approximate inference algorithm run on the parameters resulting from the surrogate likelihood can perform better than the algorithm run on true parameters.

Another related area of work is known as "Structured Learning" (Taskar et al., 2004). This is oriented towards MAP inference, rather than marginalization, and is focused on model defects– the objective maximized is given directly in terms of the the maxima of the model, rather than an indirect measure such as the likelihood. Structured learning usually does not focus on computational issues– in a general graph the MAP estimate remains difficult to find. Kulesza and Pereira (2008) provide an analysis of these issues and emphasize the importance of considering the inference process in learning.

7 Discussion

Classifiers are often learned by minimizing a loss function closely related to empirical risk. Conversely, graphical models are usually learned by optimizing scores (e.g. the likelihood) rather remote from the performance of the system in the inference stage. This paper presents an approach for learning to infer marginals through a direct minimization of empirical risk, where the risk measures the difference between the true marginals and the marginal predictions of the inference process.

There are several open questions, and promising areas for future research.

In some cases, the marginal beliefs will be used only to predict the maximum probability value for each variable. In such a case, a loss function that penalizes the resulting "labelwise" classification errors (Gross et al., 2007) could be more appropriate than the losses used here.

One advantage of this approach is that essentially the same learning algorithm could be used in the presence of hidden variables. If there are some x_i that are not observed, the sum over the variables in $L_{\log}$ or L_{quad} can simply be taken over the *observed* variables. This is potentially a significant advantage, since maximum likelihood learning of graphical models requires more advanced methods in the presence of hidden variables.

It is not clear the degree to which constraining the weights for the $f = b \log b$ to be positive term harms the flexibility of the model. One could possibly get more flexibility by only constraining F to be convex over the set of locally consistent marginals, rather than all marginals as done here (Heskes, 2006).

Another possible improvement would be to impose tighter bounds on the marginal polytope than local consistency (Sontag and Jaakkola, 2008), both in the learning and inference step.

Though the generic optimization method used here is reasonably fast, it may be possible to minimize F more efficiently by deriving a message passing algorithm.

Finally, future work could consider generalizing the function F. Provided that F is convex and continuous, the results from Section 3.3 will enable learning.

8 Acknowledgments

I think my advisor, Yiannis Aloimonos, as well as Alap Karapurkar and the anonymous reviewers for helpful comments.

References

P. Domingos. Structured machine learning: Ten problems for the next ten years. In *International Conf. on Inductive Logic Programming*, 2007.

S. Gross, O. Russakovsky, C. Do, and S. Batzoglou. Training conditional random fields for maximum labelwise accuracy. In *Advances in Neural Information Processing Systems (NIPS 2006)*, 2007.

T. Heskes. Convexity arguments for efficient minimization of the bethe and kikuchi free energies. *J. Artif. Intell. Res.*, 26:153–190, 2006.

S. Kakade, Y. Teh, and S. Roweis. An alternate objective function for markovian fields. In *International Conf on Machine Learning*, 2002.

A. Kulesza and F. Pereira. Structured learning with approximate inference. In *Advances in Neural Information Processing Systems (NIPS 2007)*, 2008.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conf. on Machine Learning*, 2001.

D. Sontag and T. Jaakkola. New outer bounds on the marginal polytope. In *Advances in Neural Information Processing Systems (NIPS 2007)*, 2008.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems (NIPS 2003)*, 2004.

S. V. N. Vishwanathan, N. Schraudolph, M. Schmidt, and K. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *International Conf. on Machine Learning*, 2006.

M. Wainwright. Estimating the "wrong" graphical model: Benefits in the computation-limited setting. *J. Mach. Learn. Res.*, 7:1829–1859, 2006.

M. Wainwright and M. Jordan. Variational inference in graphical models: The view from the marginal polytope. In *Allerton Conf. on Control, Communication and Computing*, 2003.

J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

A Proof

Claim 2: Define $\mathbf{b}^*(\boldsymbol{\theta}) \doteq \arg\min_{\mathbf{b}} F(\mathbf{b}, \boldsymbol{\theta})$, such that $A\mathbf{b} = d$ for some convex function F. Then,

$$\frac{\partial \mathbf{b}^*(\boldsymbol{\theta})}{\partial \theta_j} = (D^{-1}A^T(AD^{-1}A^T)^{-1}AD^{-1} - D^{-1})\frac{\partial^2 F}{\partial \mathbf{b}\partial\theta_j},$$

where $D = (\frac{\partial^2 F}{\partial \mathbf{b}\partial \mathbf{b}^T})$.

Proof of Claim 2: First, create a Lagrangian, enforcing the constraint.

$$\mathcal{L}(\mathbf{b}, \boldsymbol{\theta}) = F(\mathbf{b}, \boldsymbol{\theta}) + \boldsymbol{\lambda}^T(A\mathbf{b} - \mathbf{d}). \tag{28}$$

Now, apply Claim 1 to the function $\mathcal{L}$ with respect to variables $\mathbf{b}$ and $\boldsymbol{\theta}$. This gives the system

$$\begin{bmatrix} \frac{\partial \mathbf{b}}{\partial \theta_j} \\ \frac{\partial \boldsymbol{\lambda}}{\partial \theta_j} \end{bmatrix} = -\begin{bmatrix} (\frac{\partial^2 \mathcal{L}}{\partial \mathbf{b}\partial \mathbf{b}^T}) & (\frac{\partial^2 \mathcal{L}}{\partial \mathbf{b}\partial \boldsymbol{\lambda}^T})^T \\ (\frac{\partial^2 \mathcal{L}}{\partial \mathbf{b}\partial \boldsymbol{\lambda}^T}) & (\frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{\lambda}\partial \boldsymbol{\lambda}^T}) \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial \mathbf{b}\partial \theta_j} \\ \frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{\lambda}\partial \theta_j} \end{bmatrix} \tag{29}$$

The first submatrix of second partial derivatives is the same as those for the unconstrained system.

$$(\frac{\partial^2 \mathcal{L}}{\partial \mathbf{b}\partial \mathbf{b}^T}) = (\frac{\partial^2 F}{\partial \mathbf{b}\partial \mathbf{b}^T}) \doteq D \tag{30}$$

The other two matrices are constant.

$$(\frac{\partial^2 \mathcal{L}}{\partial \mathbf{b}\partial \boldsymbol{\lambda}^T}) = A \qquad (\frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{\lambda}\partial \boldsymbol{\lambda}^T}) = 0 \tag{31}$$

The first derivatives are also easy to calculate.

$$\frac{\partial^2 \mathcal{L}}{\partial \mathbf{b}\partial \theta_j} = \frac{\partial^2 F}{\partial \mathbf{b}\partial \theta_j} \qquad \frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{\lambda}\partial \theta_j} = \mathbf{0} \tag{32}$$

Substituting all this yields the system

$$\begin{bmatrix} \frac{\partial \mathbf{b}}{\partial \theta_j} \\ \frac{\partial \boldsymbol{\lambda}}{\partial \theta_j} \end{bmatrix} = -\begin{bmatrix} D & A^T \\ A & 0 \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial^2 F}{\partial \mathbf{b}\partial \theta_j} \\ 0 \end{bmatrix}. \tag{33}$$

It is possible to recover $\frac{\partial \mathbf{b}}{\partial \theta_j}$ directly from this equation. However, the form derived below can be significantly faster.

Now, consider the inverse of the above matrix. If its entries are X, Y, Z and U, by definition it satisfies

$$\begin{bmatrix} D & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} X & Y \\ Z & U \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}. \tag{34}$$

Two of the four identities that follow directly from Eq. 34 are

$$DX + A^T Z = I, \tag{35}$$

$$AX = 0. \tag{36}$$

Solving Eqs. 35 and 36 for X gives

$$X = D^{-1} - D^{-1}A^T(AD^{-1}A^T)^{-1}AD^{-1}. \tag{37}$$

The claim follows from the observation that $\frac{\partial \mathbf{b}}{\partial \theta_j} = -X\frac{\partial^2 F}{\partial \mathbf{b}\partial \theta_j}$.□

Projected Subgradient Methods for Learning Sparse Gaussians

John Duchi
Computer Science Dept.
Stanford University
Stanford, CA 94305

Stephen Gould
Electrical Engineering Dept.
Stanford University
Stanford, CA 94305

Daphne Koller
Computer Science Dept.
Stanford University
Stanford, CA 94305

Abstract

Gaussian Markov random fields (GMRFs) are useful in a broad range of applications. In this paper we tackle the problem of learning a sparse GMRF in a high-dimensional space. Our approach uses the ℓ_1-norm as a regularization on the inverse covariance matrix. We utilize a novel projected gradient method, which is faster than previous methods in practice and equal to the best performing of these in asymptotic complexity. We also extend the ℓ_1-regularized objective to the problem of sparsifying entire blocks within the inverse covariance matrix. Our methods generalize fairly easily to this case, while other methods do not. We demonstrate that our extensions give better generalization performance on two real domains—biological network analysis and a 2D-shape modeling image task.

1 Introduction

A key challenge in graphical models is learning model structure and variable independencies from data. We consider this problem in the context of Gaussian distributions defined over undirected graphs, or Gaussian Markov Random Fields (GMRFs). GMRFs are an important class of graphical models that have found applications in a wide range of areas including oceanography, image denoising, speech recognition, and FMRI (Willsky, 2002; Bilmes, 2000). In a GMRF, independencies—edges absent from the network—correspond to zero entries in the inverse covariance matrix K (Lauritzen, 1996); that is, the GMRF has an edge between variables i and j if and only if $K_{ij} \neq 0$. Thus, learning a sparse GMRF corresponds precisely to learning an inverse covariance with many zero entries.

Sparsity in the inverse covariance matrix has a number of advantages. First, promoting sparsity when learning from limited data has been shown to produce robust models that generalize well to unseen data (Dempster, 1972). Second, the cost of both exact and approximate inference (e.g., belief propagation message passing (Malioutov et al., 2006)) depends strongly on the density of the GMRF structure. Finally, the number of parameters that need to be stored can be an important computational factor in some settings. These latter issues are particularly relevant in systems, such as speech recognition (Bilmes, 2000) or tracking (Bar-Shalom & Fortmann, 1988), that need to achieve real-time performance. Sparsity can also be beneficial in a knowledge discovery setting, where the structure of the learned model may provide insight into the relationships between the variables. For example, sparse Gaussian models have been successfully used to explore interactions between the genes in gene expression data (Dobra et al., 2004).

The idea of learning sparse GRMFs goes back to the work of Dempster (1972) where elements of the inverse covariance matrix are explicitly set to zero and the remaining parameters learned from data. Other early work used a greedy forward/backward search over edges in a Gaussian MRF that quickly became infeasible as n, the dimension of the problem, grew (Lauritzen, 1996). More recently, Banerjee et al. (2006) formulated the problem as one of optimizing a log-likelihood objective regularized with an ℓ_1-norm penalty on the entries. This penalty is known (Tibshirani, 1996) to push parameters to zero, inducing sparsity. (See Section 2 for more details on this and other other approaches for learning sparse GMRFs.)

In this paper, we present a new algorithm for optimizing the ℓ_1-penalized log-likelihood objective for Gaussian distributions. Our approach is based on projected gradient methods originally proposed by Levitin and Polyak (1966). Like other methods based on this objective, our method exploits the fact that the objective is tractable and convex, hence avoiding problems from greedy or heuristic searches. However, unlike the work of Banerjee et al. (2006), the complexity of our algorithm grows as $O(n^3)$ rather than $O(n^4)$ per iteration. For problems of high dimension, this order-of-magnitude reduction in complexity can be significant.

We also generalize the sparse inverse covariance problem to that of estimating an inverse covariance with block sparsity. This task corresponds to finding a Gaussian MRF in which certain groups of edges should all be penalized

together. For example, in a GMRF over genes, we may want to jointly penalize any interaction between genes in two pathways; this penalty intuitively tries to reduce inter-pathway interactions, but once the two pathways are allowed to interact via one pair of genes, the penalty is removed for all other pairs. To our knowledge, this extension has not been considered elsewhere. Indeed, previous methods do not naturally handle this setting, whereas our method handles it easily, with no real increase in computational cost. We present results demonstrating the value of the block structure to two distinct applications: learning networks over genes from gene expression data, and learning models of the 2D shape of mammals.

2 Background and Related Work

As mentioned above, the literature on estimating covariance matrices from data has a long history. Our work starts from the formulation of Banerjee et al. (2006) which we review below. We then compare this approach to several other recent works.

2.1 ℓ_1-Regularized Problem Formulation

We assume that we are given a dataset $\mathcal{D} = \{x[1], \ldots, x[m]\}$ in which the samples $x \in \mathcal{D}$ are drawn from some n-dimensional Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. Given a Gaussian with mean μ and covariance Σ, the average log-likelihood (modulo unnecessary constants) for $\mathcal{D}$ with same mean can be written as

$$\log\det K - \operatorname{tr}(\hat{\Sigma}K)$$

where $K = \Sigma^{-1}$ is the inverse covariance matrix for the model, and $\hat{\Sigma} = \frac{1}{m}\sum_{i=1}^{m}(x[i]-\mu)^T(x[i]-\mu)$ is the empirical covariance of the dataset $\mathcal{D}$. As we discussed, the sparsity structure in the matrix K defines the structure of the GMRF corresponding to this Gaussian: a zero entry K_{ij} corresponds precisely to the absence of an edge between the variables i and j. For uniformity of exposition, we use the matrix terminology and notation in the remainder of our discussion.

Following the approach of Banerjee et al., we add sparsity-promoting ℓ_1-penalty to the entries of the inverse covariance. We can now formulate the convex optimization problem for estimating the inverse covariance of our model:

$$\underset{K\succ 0}{\text{minimize}} \quad -\log\det(K) + \operatorname{tr}(\hat{\Sigma}K) + \sum_{i,j}\lambda_{ij}|K_{ij}|, \quad (1)$$

where we use the notation $A \succ 0$ to indicate that the symmetric matrix A is positive definite. Here λ_{ij} controls the sparsity of the solution to (1). As we show in Lemma 1, so long as $\lambda_{ij} > 0$ for all $i \neq j$ (and $\lambda_{ii} \geq 0$) the solution to (1) is unique and positive definite. Furthermore, if λ_{ij} is large enough, we will force all off-diagonal entries of K to zero.

It will be instructive now and for later development in the paper to take the dual of (1). We introduce an auxiliary variable $Z = K$ and its associated dual variable $W \in \mathbb{R}^{n\times n}$, leading to the Lagrangian

$$\begin{aligned}\mathcal{L}(K,Z,W) &= -\log\det(K) + \operatorname{tr}(\hat{\Sigma}K) \\ &\quad + \sum_{i,j}\lambda_{ij}|Z_{ij}| + \operatorname{tr}(W(K-Z)).\end{aligned}$$

The above Lagrangian is separable into terms involving K and terms involving Z, which enables us to find the dual. For the terms involving Z we have that

$$\inf_Z \sum_{ij}\lambda_{ij}|Z_{ij}| - \operatorname{tr}(WZ) = \begin{cases} 0 & \text{if } |W_{ij}| \leq \lambda_{ij} \\ -\infty & \text{otherwise.}\end{cases}$$

The infimum over K for the remaining terms can be found by using $\nabla_K \log\det(K) = K^{-1}$ (and assuming that $\hat{\Sigma} + W \succ 0$) to give

$$\inf_K[-\log\det(K) + \operatorname{tr}((\hat{\Sigma}+W)K)] = \log\det(\hat{\Sigma}+W) + n$$

Combining, we get the following dual problem for (1):

$$\begin{array}{ll}\text{maximize} & \log\det(\hat{\Sigma}+W) \\ \text{subject to} & |W_{ij}| \leq \lambda_{ij} \ \ \forall i,j.\end{array} \quad (2)$$

The constraint that $\hat{\Sigma} + W \succ 0$ is implicit in the objective because of the convention that $\log\det(X) = -\infty$ when $X \not\succ 0$. We denote the set $\{W : |W_{ij}| \leq \lambda_{ij}\ \forall i,j\}$ by B_λ, which is a box constraint indexed by the tuple $\lambda = (\lambda_{ij})$. The duality gap given a dual feasible point $W \in B_\lambda$ (and hence the primal point $K = (\hat{\Sigma}+W)^{-1}$) is

$$\eta = \operatorname{tr}(\hat{\Sigma}K) + \sum_{ij}\lambda_{ij}|K_{ij}| - n.$$

Based on the dual problem, we prove the following lemma, which guarantees a solution to problem (1):

Lemma 1. *With probability 1, so long as $\lambda_{ij} > 0$ for all $i \neq j$ and $\lambda_{ii} \geq 0$, the minimization problem (1) is bounded below and has a unique optimal point $K^\star$.*

Proof We consider the case when $\lambda_{ii} = 0$ as Banerjee et al. (2006) already consider the case when $\lambda_{ii} > 0$. First, with probability 1, $\hat{\Sigma}$ has non-zero diagonals because it was generated from a set of data samples. If there is a dual feasible point for (2), i.e., some W such that $\operatorname{diag}(W) = 0$ and $|W_{ij}| \leq \lambda_{ij}$, the dual problem has a non-infinite value, and so the primal is bounded below. Further, if we can find a dual-feasible W such that $|W_{ij}| < \lambda_{ij}$ (i.e. W is in the relative interior of the domain), then Slater's constraint qualification (Boyd & Vandenberghe, 2004) guarantees a primal-dual optimal pair $K^\star = (\hat{\Sigma}+W^\star)^{-1}$ with zero duality gap. So we simply show that a W in the relative interior of the domain exists.

Consider the matrix $D = \operatorname{diag}(\hat{\Sigma}) \succ 0$. For $\alpha \in [0,1)$ we have $\alpha\hat{\Sigma} + (1-\alpha)D \succ 0$. But, for any α, we have that $\operatorname{diag}(\alpha\hat{\Sigma} + (1-\alpha)D) = \operatorname{diag}(\hat{\Sigma})$. Thus, we choose

$$W = \alpha\hat{\Sigma} + (1-\alpha)\operatorname{diag}(\hat{\Sigma}) - \hat{\Sigma}, \quad (3)$$

which has zeros on the diagonal, and for α close enough to 1 will have $|W_{ij}| < \lambda_{ij}$. Further,

$$\hat{\Sigma} + W = \alpha\hat{\Sigma} + (1-\alpha)\operatorname{diag}(\hat{\Sigma}) \succ 0.$$

Choosing W as in (3) gives a dual feasible point in the relative interior of the domain. $W^\star$ is unique by the strong convexity of $\log\det$ over the positive definite cone. □

If we can find the maximizing $W^\star$ for (2), we can also easily calculate the sparsity for $K^\star = (\hat{\Sigma} + W^\star)^{-1}$ by the complementarity conditions on W_{ij} and K_{ij} (Boyd & Vandenberghe, 2004). Specifically, if $|W^\star_{ij}| < \lambda_{ij}$, then $K^\star_{ij} = 0$. Lastly, as shown in the analysis of the so-called maximum determinant completion problem (see Boyd and Vandenberghe (2004)), if we constrain only the diagonal elements of W and do not constrain its off-diagonals, then maximizing $\log\det(\hat{\Sigma}+W)$ will drive the off-diagonal entries of K to 0; as such, choosing $\lambda_{ij} = |\hat{\Sigma}_{ij}|$ forces complete sparsity in K and we find an MRF with no edges, i.e., all the variables are independent. Thus, when solving (1), we have a natural upper bound on λ beyond which no additional sparsity can be gained.[1]

2.2 Prior Work

One recent approach for learning sparse GMRFs is to focus on learning the set of neighbors for a particular variable by regressing that variable against the remaining variables in the network with an ℓ_1-penalty to promote sparsity (Meinshausen & Bühlmann, 2006). An alternative approach, first presented by Banerjee et al. (2006), is to solve the optimization problem (2). It is convex and can be solved using interior point methods in $O(n^6\log(1/\varepsilon))$ time to a desired accuracy ε, however, this becomes infeasible for even moderate n. Banerjee et al.'s approach was to solve problem (2) iteratively, column by column, as a sequence of QPs. That is, they define $U = \hat{\Sigma} + W$ and partition U and $\hat{\Sigma}$ as

$$U = \begin{bmatrix} U_{11} & u_{12} \\ u_{12}^T & u_{22} \end{bmatrix} \quad \hat{\Sigma} = \begin{bmatrix} S_{11} & s_{12} \\ s_{12}^T & s_{22} \end{bmatrix}.$$

They then solve a sequence of box-constrained QPs (swapping rows of U to get all the rows), each iteration setting $\hat{u}_{12} = \operatorname{argmin}_y \left\{ y^T U_{11}^{-1} y \mid \|y - s_{12}\| \le \lambda \right\}$, updating U and W appropriately. This method takes $O(Tn^4)$ time, where T is the number of passes through the columns of the matrix U.

An alternative to the box-constrained QP above (its dual) is to minimize a sequence of re-weighted LASSO problems. In work developed in parallel and independently of our paper, Friedman et al. (2007) use this to solve problem (2). Their method enjoys better performance (around $O(Tn^3)$, though no analysis is given) than Banerjee et al.'s, which is similar to the methods we develop in our paper. They, however, explicitly rely on λ_{ii} being positive to get a feasible starting point when the number of samples is less than the dimension, which we do not.

We have already mentioned Meinshausen and Bühlmann's work (2006) regressing single variables against one another. The advantage of their approach is in its efficiency, as solving n regressions each with n variables can be done very quickly, i.e., in the time it takes to do one pass over all the columns with LASSO. However, they do not directly obtain a maximum likelihood estimate from their method, only the structure of the graph.

The SPICE algorithm of Rothman et al. (2007) solves (1) when $\lambda_{ii} = 0$ (i.e., applying an ℓ_1-penalty only to off diagonal elements of the inverse covariance). The advantages of this penalty are many: Rothman et al. prove that such a setting of the regularization terms is consistent so that the solution to (1) approaches the true inverse covariance as the number of data samples increases, and they give experiments showing that not penalizing the diagonal consistently learns model structure more accurately than does regularizing each entry of K. Further, Meinshausen (2005) gives simple conditions under which penalizing the on-diagonal entries of K in (1) gives the *wrong* inverse covariance, even in the limit of infinite data. These results underscore the importance of Lemma 1, which gives a dual-feasible starting point even when $\lambda_{ii} = 0$. To our knowledge, this has not been noted before this work.

The disadvantage of the SPICE algorithm is that, like Banerjee et al.'s algorithm, it performs coordinate-wise updates (though it uses the columns of the Cholesky decomposition of K). It repeatedly iterates through all the columns, solving n $O(n^3)$ regressions at every step, giving it a time complexity again of $O(Tn^4)$. Further, Rothman et al.'s algorithm has no explicit way to check convergence, as it does not generate dual-feasible points (which our algorithms do), and they rely on truncation of near-zero values rather than explicit conditions for sparsity in the inverse covariance (such as complementarity of primal-dual variables). The algorithm they propose is also somewhat complicated.

3 Projected Gradient Method

We propose a projected gradient method for solving the dual problem (2). Projected gradient algorithms minimize an objective $f(x)$ subject to the constraint that $x \in S$ for some convex set S.[2] They do this by iteratively updating

$$x := \Pi_S(x + t\nabla f(x))$$

where t is a step size and $\Pi_S(z) = \operatorname{argmin}_y\{\|z-y\|_2 \mid y \in S\}$ is the Euclidean projection onto set S (Bertsekas, 1976). First order projected gradient algorithms are effective when second order methods are infeasible because of the dimension of the problem. As the dimensions of our problems

[1] Presumably one could regularize the diagonals further, though this seems of little practical use.

[2] In our case S is the set of $n \times n$ matrices in the box B_λ.

Algorithm 1 Maximize $\log\det(\hat{\Sigma}+W)$ subject to $W \in B_\lambda = \{W \mid |W_{ij}| \le \lambda_{ij}\}$. Given empirical covariance $\hat{\Sigma} \succeq 0$, λ_{ij}, duality gap stopping criterion ϵ, and W such that $W \in B_\lambda$, $\hat{\Sigma}+W \succ 0$, and $W_{ii} = \lambda_{ii}$.

1: **repeat**
2: *Compute unconstrained gradient*
3: $G := (\hat{\Sigma}+W)^{-1}$
4: *Zero components of gradient which would result in constraint violation*
5: $G_{ii} := 0$
6: $G_{ij} := 0$ for all $W_{ij} = \lambda_{ij}$ and $G_{ij} > 0$
7: $G_{ij} := 0$ for all $W_{ij} = -\lambda_{ij}$ and $G_{ij} < 0$
8: *Perform line search*
9: $t :\approx \operatorname{argmax}_t \log\det(\hat{\Sigma} + \Pi_{B_\lambda}(W+tG))$
10: *Update and project*
11: $W := \Pi_{B_\lambda}(W+tG)$
12: $K := (\hat{\Sigma}+W)^{-1}$
13: *Compute duality gap*
14: $\eta = \operatorname{tr}(\hat{\Sigma}K) + \sum_{ij}\lambda_{ij}|K_{ij}| - n$
15: **until** $(\eta < \epsilon)$ or maximum iterations exceeded
16: **return** K

Algorithm 2 Line search to find feasible t for $\log\det(\hat{\Sigma}+W+tG)$ given $f_0 = \log\det(\hat{\Sigma}+W)$.

1: $t := \frac{\operatorname{tr}((\hat{\Sigma}+W)^{-1}G)}{\operatorname{tr}((\hat{\Sigma}+W)^{-1}G(\hat{\Sigma}+W)^{-1}G)}$
2: **while** $\log\det\left(\hat{\Sigma} + \Pi_{B_\lambda}(W+tG)\right) \le f_0$ **do**
3: $t := t/2$
4: **end while**

often exceed $n = 1000$, giving more than 500,000 different parameters, this makes projected gradient methods a reasonable choice.

The projected gradient method for our problem is shown in Algorithm 1. The unconstrained gradient of the dual objective function (2) is $G = (\hat{\Sigma}+W)^{-1}$. We perform a line search to find the step size t that approximately gives the greatest increase to the objective. This search needs to guarantee that the estimated covariance $\hat{\Sigma}+W$ is positive definite, which it does because we assume that $\log\det X = -\infty$ for $X \not\succ 0$. We can guarantee that the initial $\hat{\Sigma}+W \succ 0$, because $\hat{\Sigma} \succeq 0$, and we can simply initialize $W_{ii} = \lambda_{ii}$ and the rest of W via Lemma 1 (this initialization is optimal for the diagonal elements, so we do not modify them through the course of the algorithm). The projection of the gradient appears in two places. First, we immediately zero out some entries of the gradient when W_{ij} is at the boundary and G_{ij} would push W_{ij} outside B_λ. Second, during the line search, in each step we project the gradient onto the box-constraint $|W_{ij}| \le \lambda_{ij}$ via the operation $\Pi_{B_\lambda}(W+tG)$, which simply sets any entry $> \lambda_{ij}$ to λ_{ij} (and likewise for $-\lambda_{ij}$).

For the simple box-constrained projections, because we can immediately zero out many entries of the gradient G and still have a descent direction (see lines 5 through 7 of Algorithm 1, which are not strictly necessary but improve the performance of the line search), a simple heuristic line search based on the second-order approximation to $\log\det$ performs very well. The second order expansion of the log-determinant function around a point X (Boyd & Vandenberghe, 2004) is given by $\log\det(X+\Delta X) \approx \log\det(X) + \operatorname{tr}(X^{-1}\Delta X) - \frac{1}{2}\operatorname{tr}(X^{-1}\Delta X X^{-1}\Delta X)$. Thus, given the descent direction G, we approximate $\log\det(\hat{\Sigma}+W+tG)$ by

$$\log\det(\hat{\Sigma}+W) + t\operatorname{tr}((\hat{\Sigma}+W)^{-1}G) - \tfrac{1}{2}t^2\operatorname{tr}((\hat{\Sigma}+W)^{-1}G(\hat{\Sigma}+W)^{-1}G)$$

and perform the line search of Algorithm 2. Convergence is guaranteed because the iterates for W form a sequence in a compact space B_λ and $\log\det(\hat{\Sigma}+W)$ is always increasing. If the line search cannot return a satisfying t, then no improvement can be made in the projected descent direction, so standard arguments by KKT conditions for a differentiable convex function guarantee that we have reached the optimum.

4 Structure Extensions

We can extend the basic problem of (1) to cases in which we are interested in sparsity not just between single variables but between entire blocks of variables. In many problem domains, variables can be naturally grouped into blocks. For example, we might try to model a 2D shape made up of articulated objects (such as the outline of an animal) in which we want to regularize interactions between object parts (such as legs, body, head, and tail). Landmarks along the contour of an animal's head can naturally be grouped together, as these landmarks move collectively as the animal moves through different articulated forms. In modeling gene networks, we may want to encode the intuition that interactions happen at the level of pathways, i.e., either two pathways interact, in which case multiple genes can be involved, or they do not interact at all. Block regularization might also arise in the context of learning multi-resolution models constrained so that only variables within specific resolution levels interact (Willsky, 2002).

In the block regularization case, we let the entries in our inverse covariance matrix be divided into $p < n^2$ disjoint subsets $S_1, \ldots, S_p$ (the disjointness assumption is essential to our method, as we see below). We can find the dependencies between the subsets by solving the following block ℓ_1-regularized log likelihood problem:

$$\begin{aligned}\text{minimize} \quad & -\log\det(K) + \operatorname{tr}(\hat{\Sigma}K) \\ & + \sum_k \lambda_k \max\{|K_{ij}| : (i,j) \in S_k\}.\end{aligned} \tag{4}$$

The subset S_k encodes a set of interactions that, as soon as one of the interactions in the set exists, there is no reason

to penalize any of the other interactions. For example, in the 2D shape model, the subsets S_k may be constructed as $S_{qr} = \mathcal{B}_q \times \mathcal{B}_r$ where the $\mathcal{B}_q$ and $\mathcal{B}_r$ represent the set of variables belonging to the q^{th} and r^{th} articulated body part, respectively.

We can perform a similar derivation to the one in Section 2.1 to find the dual problem for (4):

$$\begin{array}{ll} \text{maximize} & \log\det(\hat{\Sigma} + W) \\ \text{subject to} & \sum_{(i,j)\in S_k} |W_{ij}| \leq \lambda_k, \quad \text{for } k = 1, \ldots, p. \end{array} \tag{5}$$

To apply a projected gradient method to (5), we need to project to the constraints $\sum_{(i,j)\in S_k} |W_{ij}| \leq \lambda_k$. Recent work by Duchi et al. (2008) provides such a method, giving a randomized linear time algorithm to project a vector w to the constraint $\|x\|_1 \leq C$. The sets S_k must be disjoint so that we can project to each constraint in (5) independently (and hence efficiently). We refer the reader to their paper for a description of the actual projection algorithm.

Given the linear time algorithm to project to an ℓ_1 constraint, we can develop an $O(n^2)$ expected time (the same efficiency as the projection to B_λ of earlier methods) method to project to the feasible set $S = \{W \mid \sum_{(i,j)\in S_k} |W_{ij}| \leq \lambda_k, k = 1, \ldots, p\}$. The method simply iterates through each of the blocks S_k, projecting $W_{ij} : (i,j) \in S_k$ to the constraint that $\sum_{(i,j)\in S_k} |W_{ij}| \leq \lambda_k$ to get $W \in S$. With this as a building block, Algorithm 3 maximizes the dual (5). Because the constraint set is more complicated, making it difficult to make the second order expansion of $\log\det$ accurate for the projected direction, the algorithm uses Armijo-like line searches as described by Bertsekas (1976) rather than the simpler line search of Algorithm 1. If we let $g = \nabla f(x)$, the Armijo line search for a projected gradient maximization method with function f returns the first t such that

$$f(\Pi_S(x + tg)) \geq f(x) + \alpha \nabla f(x)^T (\Pi_S(x + tg) - x)$$

where t is initialized to 1 and decreased by a multiplier $\beta < 1$ every time the above condition is not satisfied. In our algorithm t is adaptively chosen to make the line search converge more quickly (see line 22 of Alg. 3), and setting $0 < \alpha < 1$ guarantees convergence of the method.

Previous methods developed for solving the penalized maximum-likelihood problem for Gaussians are unable to handle the block ℓ_1-penalties of (4). Effectively, batch steps are needed to handle the more long range block constraints; the coordinate based methods of Banerjee et al. (2006), Friedman et al. (2007), and Rothman et al. (2007) cannot account for more global constraints that tie parameters in arbitrary columns and rows. Our projected gradient methods, however, extend to the case of block penalties, with ease. Further, the added complexity is negligible, as the entire projection step is still $O(n^2)$, while the expensive $O(n^3)$ step is computing the gradient.

Algorithm 3 Solves (5) given an initial W such that $\hat{\Sigma} + W \succ 0$ and constants β and α, $0 < \beta < 1$ and $0 < \alpha < 1$.

1: $t = 1$
2: **repeat**
3: *Compute unconstrained gradient*
4: $G = (\hat{\Sigma} + W)^{-1}$
5: *Compute direction of step*
6: $D = \Pi_S(W + tG) - W$
7: *Compute initial and next objective values*
8: $f_0 = \log\det(\hat{\Sigma} + W)$
9: $f_t = \log\det(\hat{\Sigma} + \Pi_S(W + tG))$
10: *Perform backtracking line search*
11: **while** $f_t < f_0 + \alpha \operatorname{tr}(DG)$ **do**
12: *Decrease t, recalculate direction and objective*
13: $t = \beta \cdot t$
14: $D = \Pi_S(W + tG) - W$
15: $f_t = \log\det(\hat{\Sigma} + \Pi_S(W + tG))$
16: **end while**
17: *Compute next points and duality gap*
18: $W = \Pi_S(W + tG)$
19: $K = (\hat{\Sigma} + W)^{-1}$
20: $\eta = \operatorname{tr}(\hat{\Sigma}K) + \sum_k \lambda_k \|K_{ij} : (i,j) \in S_k\|_\infty - n$
21: *Increase t slightly*
22: $t = t/\beta$
23: **until** $\eta < \epsilon$

5 Experimental Results

In this section, we describe our experimental results. We performed experiments both on synthetic and real data, gathering timing information as well as calculating log-likelihood on test data, validating the usefulness of sparse estimators. This validation seems to have been notably absent from much of the literature on sparse inverse covariance selection.

5.1 Timing Results

We ran a series of timing experiments for the original problem from equations (1) and (2) comparing our approach to that of Banerjee et al. (2006) and Friedman et al. (2007). To generate data for our timing experiments, we constructed random $n \times n$ sparse inverse covariance matrices with roughly 20 edges per node. For each such matrix, we generated $n/3$ samples and used them to construct the empirical covariance $\hat{\Sigma}$. We selected λ for the penalty in (1) so that at solution, the inverse covariance K had (approximately) the correct number of edges. Our timing experiments were run on a computer with a 1.7Ghz Intel Xeon 32-bit processor and 1.96GB of RAM. The run times to achieve a duality gap of $\epsilon = 0.1$ are plotted in Fig. 1, which shows CPU time versus problem size compared to Banerjee et al.'s and Friedman et al.'s column-wise coordinate ascent algorithms. The results show that our projected subgradient method outperforms Banerjee et al.'s column-wise ascent by one to two orders of magnitude on these

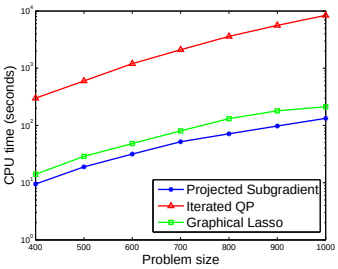


Figure 1: Run times in log-scale of the method presented in this paper versus Banerjee et al.'s (Iterated QP) and Friedman et al.'s (Graphical Lasso).

tests. Our method seems to find a solution in roughly half to two-thirds the time that Friedman et al.'s does, though their method seems to more easily be optimized when the primal matrix K becomes sparse. Further, in our experiments when $\hat{\Sigma}$ was not full rank, Friedman et al.'s method seemed to have more difficulty maintaining a positive definite solution than did ours, which guarantees positive definiteness of $\hat{\Sigma}+W$ throughout. We note that due to the slightly more complicated projection, the run times of Alg. 3 are roughly twice those of Alg. 1.

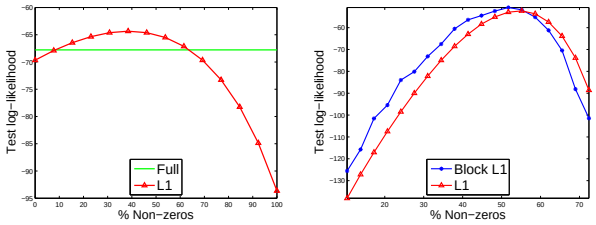
5.2 Synthetic Log Likelihoods

To test the impact of learning sparse structures, we compared our ℓ_1-penalized inverse covariance estimation from (1) to Tikhonov regularized covariance matrices, i.e., selecting the covariance matrix to be $\Sigma = \hat{\Sigma} + \nu I$ for some $\nu > 0$, which guarantees the positive-definiteness of Σ. To do this, we randomly generated 20 inverse covariance matrices with 50% sparsity and dimension $n = 60$. From each of these, we generated 30 samples for a training set (defining a covariance $\hat{\Sigma}$) and 30 samples for a test set. We then varied ν to compute the best *test set* log-likelihood achieved by Tikhonov regularized covariance across all ν's. Fig. 2(a) shows this best-case test log-likelihood for the Tikhonov regularized covariance versus the log-likelihood for sparse covariances matrices output by Algorithm 1 as we sweep the penalty parameter λ from $\max(|\hat{\Sigma}_{ij}|)$ to 10^{-3}. The results show that, for appropriate levels of sparsity, ℓ_1-regularized covariance estimation outperforms simple regularized estimates of the full covariance.

5.3 Mammal 2D Shape Models

Moving on to real data, we aimed to study whether our block ℓ_1 approach achieves better generalization than other approaches for learning the model. In our first application, we consider a two-dimensional shape classification task. Here, we have a series of 60 landmarks, each an (x, y) point in 2D, defining the outline of a mammal as a vector in $\mathbb{R}^{120}$. The mammals are from six classes: bison, deer, elephants, giraffes, llamas, and rhinos. There are on average 42 examples from each class, each a hand-labeled outline from a real image. Our task is to model each animal's outline as a Gaussian distribution, learning a mean $\mu^{(c)}$ and covariance $\Sigma^{(c)}$ for each of the six animal classes c.

We used and compared the standard sparsifying objective



(a) (b)

Figure 2: Log-likelihood results: (a) for synthetic data, comparing ℓ_1-penalized inverse covariance to (best-case) full covariance; (b) for 2D shape-recognition test data, comparing the block versus the non-block method.

(1) and the block-penalized objective (4) to learn sparse inverse covariances $K^{(c)}$. For the block penalties, we manually chose the articulated body parts $\mathcal{B}_q$ as the head, neck, stomach, each leg, back, and tail. The blocks S_k were then $S_k = \mathcal{B}_q \times \mathcal{B}_r$ for all pairs q and r (including $q = r$), and the penalty λ_k for S_k was set so that $\lambda_k \propto |S_k|$.[3]

We measure the performance of the competing methods in two ways: by test set log-likelihood and by performance on a classification task. In the log-likelihood case, we performed five-fold cross validation of the training data and swept the λ penalties for both (1) and (4), ranging from full-sparsity to no-sparsity solutions for the inverse covariance K. We set the mean $\mu^{(c)}$ for each class simply as the training set mean. As a baseline, we also chose a "full" covariance matrix $\hat{\Sigma} + \nu I$ for each class using the Tikhonov regularization technique described above. As above, we selected ν to give the best test log-likelihood for each class in our cross-validation procedure. The block sparsity led to moderate improvements in the test likelihood, which can be seen in Fig. 2(b); we note that we do not plot the full covariance result, as its best average log-likelihood on test data is -173.22, which is off the bottom of the plot. We see that ℓ_1-regularization significantly improves generalization, and that our block approach provides yet an additional improvement.

Other benefits of the block structured regularization can be seen in a classification task. In this case, for each of the three methods (sparse, block sparse, and Tikhonov regularized) and for each class we chose the inverse covariance $K^{(c)}$ that maximized the Gaussian log-likelihood on a held-out validation set. The task is to classify examples from a test set, that is, to assign a label $\hat{c}$ to a given vector $v \in \mathbb{R}^{120}$ from the test set. We assign the label for v simply as the class that has the maximum likelihood for v:

$$\hat{c} = \operatorname*{argmax}_c \left\{ \log\det K^{(c)} - (v - \mu^{(c)})^T K^{(c)} (v - \mu^{(c)}) \right\}.$$

The false positive and false negative error rates over ten different testing runs for each class are in Table 1. We

[3]Having the penalties on the blocks correspond to their sizes overcomes the tendency for larger blocks to have edges since they can more easily affect the log-likelihood.

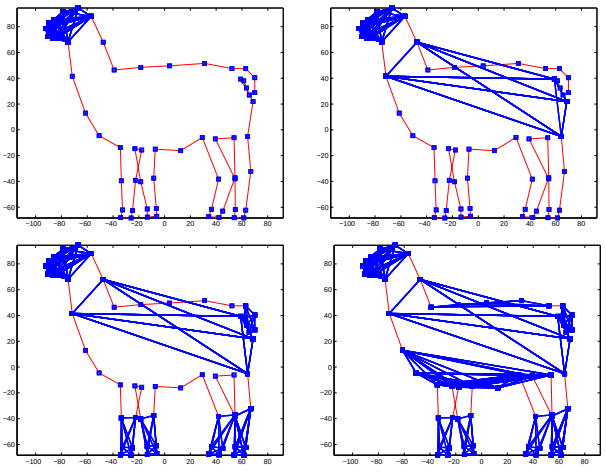


Figure 3: Different sparsity structures learned on blocked llama shape data.

Table 1: Classification Error Rates on Animals

NEG	Bison	Deer	Elephant	Giraffe	Llama	Rhino
ℓ_1	5.6%	2.2%	1.3%	5.0%	40.0%	**3.3%**
Block	**1.1%**	**0%**	**0%**	**3.0%**	**35.5%**	**3.3 %**
Full	10.0%	1.1%	2.5%	6.0%	41.1%	10.0%
POS	Bison	Deer	Elephant	Giraffe	Llama	Rhino
ℓ_1	1.2%	35.3%	**1.2%**	**0%**	3.6%	**0%**
Block	**1.1%**	**29.1%**	**1.2%**	**0%**	**0%**	**0%**
Full	2.4%	40.3%	1.3%	**0%**	1.9%	**0%**

see that the standard ℓ_1-regularized objective almost always outperforms the full inverse covariance, and that the block ℓ_1-regularized likelihood consistently and significantly outperforms both other methods.

The block regularized covariance selection can also be visualized as in Fig. 3. In the figure, we display the sparsity (i.e. the edges in the associated MRF) of the block inverse covariance learned for a llama outline as we vary the penalty term. We can see that generally, the articulated parts (legs, head, tail, body) have edges between themselves before edges between parts; intuitively, the relationships within articulated parts, such as the head or the legs, capture the shape of the part and are likely to be more informative for density estimation of the shape distribution than long range interactions such as the head's position in relation to the back right leg.

5.4 Gene Expression Data

We considered a data set that measures the mRNA expression levels of the 6152 genes in *S. cerevisiae* (baker's yeast), measured under various environmental stress conditions (Gasch et al., 2000). The expression level of the genes can be modeled as random variables and each experiment as a data sample. Some of the expression data was missing and we used a standard nearest-neighbor method to impute the missing values (Troyanskaya et al., 2001). We restricted the dataset to the 667 genes involved in known metabolic pathways (Förster et al., 2003). We preprocess the data so that each variable is zero mean and unit variance across the dataset.

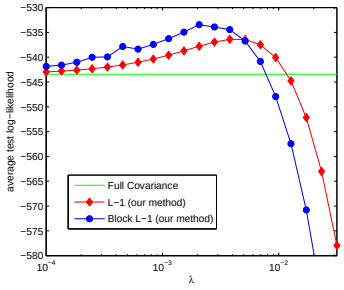


Figure 4: Average test log-likelihood on Gasch dataset for ℓ_1 (red diamonds) and block ℓ_1 (blue circles) regularization of inverse covariance matrix, compared to the full-covariance baseline (horizontal green line).

For the block experiments we group genes into 86 disjoint sets $\mathcal{B}_q$ that correspond to known metabolic pathways (Förster et al., 2003).[4] We then construct the edge subsets as follows. We have one block S_{qr} for each pair of pathways $q \neq r$, which contains all the covariance entries for $\mathcal{B}_q \times \mathcal{B}_r$. We then have a separate block S_{ij} for each pair of genes i, j in the same pathway. That is, we apply block ℓ_1 regularization between groups of genes in different blocks and standard ℓ_1 regularization between individual genes within the same block. For each block S_k we again set $\lambda_k = \lambda \cdot |S_k|$ where λ is a shared regularization parameter.

We conducted 5-fold cross-validation on the dataset. On each fold we use the training set to estimate the covariance matrix and compute the log-likelihood for each instance in the test set. We report the average log-likelihood over the 5 folds. We baseline our method by first estimating the full covariance matrix. Since the number of data samples (174) is significantly smaller than the number of variables (667), we again use cross-validated Tikhonov regularization as previously described to give an estimate for Σ. We compare our results against the best (highest log-likelihood on test) Tikhonov regularized covariance.

Fig. 4 shows the performance of our methods compared to the baseline. The results clearly illustrate that the sparsity inducing ℓ_1-regularized objective outperforms the baseline, while the block ℓ_1-regularized penalties give further benefits in terms of log-likelihood.

6 Conclusion

In this work, we have presented new methods for finding sparse inverse covariance matrices and thereby for selecting edge structures for Gaussian MRFs. The methods we present are significantly faster than prior work, and they generalize straightforwardly to learning more complicated structures than has been previously possible. We also provide compelling experimental results on two real-world

[4] A small number of genes that participate in more than one pathway were placed into individual sets of 1 gene each.

data sets demonstrating the benefits of this approach for both density estimation and classification.

Our work suggests the promise of projected gradient methods for other ℓ_1-regularized problems by optimization in the dual space, as the duals of these problems often have simple ℓ_∞ constraints to which it is trivial to project. However, this approach currently relies on the ability to efficiently recover the primal variables from the dual variables, making its general application an open problem. Another interesting direction for future work is the construction of methods for intelligently setting the penalty parameters λ. This could certainly lead to more accurate structure recoveries, and recent work by Do et al. (2007) demonstrates the promise of hyperparameter learning in log-linear models; this might be extended to more general problems such as structure learning. Our work demonstrates the benefit of block-structured regularization. However, the blocks must currently be selected by hand and cannot overlap. Automatically learning the block structure would be a very useful extension to our work, both for improved performance and for the explanatory power it gives beyond individual edge structures. Finally, ℓ_1-regularized learning has recently been demonstrated successfully for discrete MRFs (Lee et al., 2006; Wainwright et al., 2007). It would be interesting to see whether efficient projection-based methods such as ours can be applied to richer settings that involve discrete variables or non-linear continuous interactions.

Acknowledgments

We thank Stephen Boyd for his guidance and useful comments, Gal Elidan and Geremy Heitz for help with the mammals data set, Gal Chechik and Su-In Lee for help with the gene expression data set, and the anonymous reviewers for helpful feedback. This work was supported by the Office of Naval Research under MURI N000140710747, DARPA under the Transfer Learning program, and NSF grant BDI-0345474.

References

Banerjee, O., Ghaoui, L. E., d'Aspremont, A., & Natsoulis, G. (2006). Convex optimization techniques for fitting sparse gaussian graphical models. *Proceedings of the 23^{rd} International Conference on Machine Learning*.

Bar-Shalom, Y., & Fortmann, T. E. (1988). *Tracking and data association*. Academic Press.

Bertsekas, D. P. (1976). On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Trans. on Automatic Control, 21*.

Bilmes, J. A. (2000). Factored sparse inverse covariance matrices. *IEEE Conference on Acoustics, Speech and Signal Processing*.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Dempster, A. P. (1972). Covariance selection. *Biometrics, 28*, 157–175.

Do, C. B., Foo, C.-S., & Ng, A. Y. (2007). Efficient multiple hyperparameter learning for log-linear models. *Neural Information Processing Systems 21*.

Dobra, A., Hans, C., Jones, B., Nevins, J., Yao, G., & West, M. (2004). Sparse graphical models for exploring gene expression data. *Journal of Multivariate Analysis, 90*, 196–212.

Duchi, J., Shalev-Shwartz, S., Singer, Y., & Chandra, T. (2008). Efficient projections onto the ℓ_1-ball for learning in high dimensions. *Proceedings of the 25^{th} International Conference on Machine Learning*.

Förster, J., Famili, I., Fu, P., & Palsson, B. O. (2003). Genome-scale reconstruction of the saccharomyces cerevisiae metabolic network. *Genome Research, 13*, 244–253.

Friedman, J., Hastie, T., & Tibshirani, R. (2007). Sparse inverse covariance estimation with the graphical lasso. *Biostatics*, 1–10.

Gasch, A., Spellman, P., Kao, C., Carmel-Harel, O., Eisen, M., Storz, G., Botstein, D., & Brown, P. (2000). Genomic expression program in the response of yeast cells to environmental changes. *Molecular Biology of the Cell, 11*, 4241–4257.

Lauritzen, S. L. (1996). *Graphical models*. Clarendon Press.

Lee, S.-I., Ganapathi, V., & Koller, D. (2006). Efficient structure learning of Markov networks using ℓ_1-regularization. *Neural Information Processing Systems*.

Levitin, E. S., & Polyak, B. T. (1966). Constrained minimization problems. *USSR Computational Math and Mathematical Physics, 6*, 1–50.

Malioutov, D. M., Johnson, J. K., & Willsky, A. S. (2006). Walk-sums and belief propagation in gaussian graphical models. *Journal of Machine Learning Research, 7*, 2031–2064.

Meinshausen, N. (2005). A note on the lasso for Gaussian graphical model selection. Available at http://www.stats.ox.ac.uk/ meinshau/note.pdf.

Meinshausen, N., & Bühlmann, P. (2006). High dimensional graphs and variable selection with the lasso. *Annals of Statistics, 34*, 1436–1462.

Rothman, A., Bickel, P., Levina, E., & Zhu, J. (2007). *Sparse permutation invariant covariance estimation* (Technical Report 467). University of Michian Dept. of Statistics.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B, 58*, 267–288.

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., & Altman, R. (2001). Missing value estimation for DNA microarrays. *Bioinformatics, 17*, 520–525.

Wainwright, M. J., Ravikumar, P., & Lafferty, J. D. (2007). High-dimensional graphical model selection using ℓ_1-regularized logistic regression. *Neural Information Processing Systems 19*.

Willsky, A. (2002). Multiresolution markov models for signal and image processing. *Proceedings of IEEE, 90*, 1396–1548.

Knowledge Combination in Graphical Multiagent Models

Quang Duong **Michael P. Wellman** **Satinder Singh**

University of Michigan
Computer Science & Engineering
Ann Arbor, MI 48109-2121 USA
{qduong,wellman,baveja}@umich.edu

Abstract

A *graphical multiagent model* (GMM) represents a joint distribution over the behavior of a set of agents. One source of knowledge about agents' behavior may come from game-theoretic analysis, as captured by several graphical game representations developed in recent years. GMMs generalize this approach to express arbitrary distributions, based on game descriptions or other sources of knowledge bearing on beliefs about agent behavior. To illustrate the flexibility of GMMs, we exhibit game-derived models that allow probabilistic deviation from equilibrium, as well as models based on heuristic action choice. We investigate three different methods of integrating these models into a single model representing the combined knowledge sources. To evaluate the predictive performance of the combined model, we treat as actual outcome the behavior produced by a reinforcement learning process. We find that combining the two knowledge sources, using any of the methods, provides better predictions than either source alone. Among the combination methods, mixing data outperforms the opinion pool and direct update methods investigated in this empirical trial.

1 INTRODUCTION

Graphical models provide a compact representation for domains with decomposable structure, with concomitant computational advantages. Multiagent scenarios may be particularly amenable to decomposition, to the extent that interactions among the agents exhibit localized effects. The idea of exploiting conditional independence among the effects of agents' decisions was central to the multiagent influence diagram (MAID) framework developed by Koller and Milch (2003). This observation was also a driving motivation for *graphical game* models, first introduced by Kearns et al. (2001), and subsequently examined and extended in several research efforts (Kearns, 2007).

In the basic graphical game approach, the model is a factored representation of a normal-form game, and special-purpose algorithms operate on this representation to identify approximate or exact Nash equilibria. Daskalakis and Papadimitriou (2006) demonstrated how to map a graphical game to a Markov random field (MRF), assigning high potential to configurations where an agent plays a best response to its neighbors. They showed that the maximum a posteriori configurations of the MRF correspond to pure-strategy Nash equilibria (PSNE) of the game. This approach enables the exploitation of statistical inference tools for game-theoretic computation, including the full repertoire of graphical model algorithms.

We build on these works to introduce *graphical multiagent models* (GMMs), which are simply graphical models where the joint probability distribution is interpreted as an uncertain belief (e.g., a prediction) about the agents' play. For instance, the Daskalakis and Papadimitriou mapping can be viewed as a GMM where we believe that the agents will play a PSNE if one exists. This is of course just one candidate for belief based on game-theoretic analysis. When reasoning about strategies to play, or designing a mechanism (which induces a game for other agents), we may wish to adopt alternative bases for forming beliefs about the agents' play (Vorobeychik and Wellman, 2006). The GMM framework supports such decision making, and moreover, allows that beliefs may be based on variant solution concepts, models of bounded rationality or equilibrium selection, or for that matter knowledge that has nothing to do with game-theoretic analysis. At this level, our motivation shares the spirit of the network of influence diagrams formalism of Gal and Pfeffer (2008), which extends MAIDs to incorporate non-

maximizing models of behavior. It also aligns with the goal of Wolpert's information-theoretic framework for modeling bounded rationality in game play (Wolpert, 2006).

In this paper, we illustrate the flexibility of GMMs by showing how to construct plausible multiagent models using quite different sources of belief about agent play. One example model is based on the game form, and another based on heuristic assumptions of behavior. In both, we assume that the graphical structure representing interactions among players in the game is known. We then introduce and test three approaches to integrate these knowledge sources into a combined model. To evaluate the results, we posit that actual play is generated by agents who start to play heuristically, then update their behavior over repeated interactions through a reinforcement learning (RL) process. Thus, the task we set up is to predict the outcome of a RL regime. More precisely, we seek to compute a reasonable estimation of the joint probability distribution of the agents' play. Intuitively, knowledge about the heuristic starting point is relevant, as is knowledge of strategically stable policies (game-theoretic equilibria), but neither directly captures nor necessarily corresponds to the RL outcome. We find experimentally that in fact the two knowledge sources are complementary, as the combined model outperforms either alone. Our investigation further provides support for one particular combination approach, based on mixing data.

We begin with formal definitions and specifications of the GMM framework in Section 2. In Section 3, we present an example multiagent domain, the Internet industry partnership network, and construct two plausible models based on different knowledge sources. Section 4 details some alternative combination methods. We follow with an empirical study in Section 5, designed to evaluate the performance of the respective models and their combinations. We conclude the paper with some observations on these results.

2 GRAPHICAL MULTIAGENT MODELS

Consider a multiagent scenario with n players, where each player $i \in \{1, \ldots, n\}$ chooses an action (or *strategy*) s_i, from its strategy domain, S_i. The outcome is a joint action, or *strategy profile*, s, designating the strategy choice of all players. A GMM G for this scenario is a graphical model, $G = (V, E, S, \pi)$, with vertices $V = \{v_1, \ldots, v_n\}$ corresponding to the agents (we refer to v_i and i interchangeably), and edges $(i, j) \in E$ indicating a local interaction between i and j. The graph defines for each agent a neighborhood, $N_i = \{j \mid (i, j) \in E\} \cup \{i\}$, including i and its neighbors $N_{-i} = N_i \backslash \{i\}$. Each neighborhood i is associated with a potential function $\pi_i(s_{N_i}) : \Pi_{j \in N_i} S_j \rightarrow \mathbb{R}$. Intuitively a local configuration of strategies with a higher potential is more likely to be part of the global outcome than one with lower potential. As in graphical games, the size of the GMM description is exponential only in the size of local neighborhoods rather than in the total number of players. These local potentials define the joint probability of a global configuration s,

$$\Pr(s) = \frac{\Pi_i \pi_i(s_{N_i})}{Z}, \tag{1}$$

where Z is a normalization term.

One source of potential functions is a description of the *game* played by the n agents. Let $\mathbf{G}$ be a game with agents and strategy sets as defined for the GMM G. Let us further assume that $\mathbf{G}$ is a graphical game, such that agent i's payoff depends only on s_{N_i}: its strategy and those of its neighbors. Formally, i's payoff is defined by a utility function, $u_i : \Pi_{j \in N_i} S_j \rightarrow \mathbb{R}$. For example, Daskalakis and Papadimitriou (2006) defined a binary potential function, associating a high value for configurations where each agent's strategy choice is a best response to its neighbors (i.e., maximizes payoffs given $s_{N_{-i}}$), and a low value for all other configurations.

A natural generalization of this approach would smooth out the binary distinction, assigning intermediate potentials based on the *degree* to which agents deviate from their best response. We may not wish to assume that agents play best responses with certainty, as they may not be perfectly rational, or our attributions of payoff functions may be inexact. For a given payoff model, let $\epsilon_i(s_{N_i})$ denote i's *regret* function, representing the maximum gain i can obtain through unilaterally reconsidering its own strategy s_i given $s_{N_{-i}}$,

$$\epsilon_i(s_{N_i}) = \max_{s'_i \in S_i} u_i(s'_i, s_{N_{-i}}) - u_i(s_{N_i}).$$

Intuitively, we expect that high-regret profiles are less likely to be played (all else equal), since as regret increases agents are more apt to recognize and select the better alternatives. We can capture this intuition in a *regret potential*,

$$\pi_i(s_{N_i}) = e^{-\epsilon_i(s_{N_i})/T_i}, \tag{2}$$

where T_i, the *temperature* parameter, provides a way to calibrate our association between regret and relative likelihood. Greater values of T_i accord more likelihood to agents making less than perfectly rational decisions. For simplicity in notation below, we also define $\lambda_i = \frac{1}{T_i}$, and λ the vector of λ_i.

Let *reG* denote a GMM employing the regret potential function. In the current study, we consider the

regret GMM *reG* as one plausible form of predictive model. We also consider models that are not based directly on payoffs in an associated game. In particular, we construct for our particular example a rule-based GMM, *hG*, encoding heuristic assumptions of agents' behavior.

3 EXAMPLE: INTERNET INDUSTRY PARTNERSHIPS

We illustrate the GMM framework and motivate the problem of combining knowledge sources through an example multiagent scenario. In the *Internet industry partnership* domain,[1] companies must decide whether to retain ($s = 1$) or upgrade ($s = 2$) their current technology. The payoff functions in $\mathbf{G}$ can be mapped into G's potential functions in several different ways. The payoff for each strategy depends on the choices of other companies to which they are related through some kind of partnership—their neighbors in the interaction graph. For example, the benefits of upgrading may be larger when one's partners also upgrade, since keeping their technologies synchronized enhances compatibility.

3.1 GAME DEFINITION

We construct our example scenario using a fragment of the partnership network consisting of 10 representative companies, as depicted in Figure 1. Each node represents a company, which we characterize by three parameters: (1) size class, z; (2) sector, t: either commerce, infrastructure, or content; and (3) change coefficient, $ch \in [0, 1]$, representing the intrinsic adaptability of the company's technology. The study by Krebs (2002) provides sector and a rough order of size; the change coefficient is assigned by us in an arbitrary manner. Although somewhat contrived, the scenario specification serves our purpose of demonstrating some capabilities of the GMM approach.

The payoff function, u, defines the value of retaining or upgrading technology, given the actions of a firm's neighboring companies and the parameters describing these companies. Qualitatively, payoff is increased by agreement with neighbors, where larger neighbors from the same sector are relatively more important. Let us first introduce an intermediate value w_{ij} for each connected pair of companies, reflecting the strength of i and j's partnership:

$$w_{ij}(s_i, s_j) = (z_i + z_j)\left(1 + \frac{y_{ij}}{2^{I_t} + 4^{1-I_t}}\right)^{I_s}, \qquad (3)$$

[1] The example is inspired by the network model of Krebs (2002), cited by Kearns (2002).

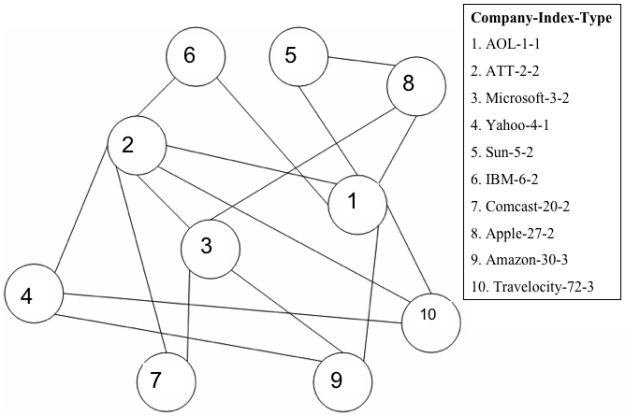


Figure 1: Part of the Internet industry partnership network, from Krebs (2002).

where $y_{ij} \sim U[0, 1]$ is a random variable, and I_s and I_t are indicator functions representing agreement in strategy and technology sector. $I_s = 1$ if $s_i = s_j$, and 0 otherwise; $I_t = 1$ if $t_i = t_j$, and 0 otherwise. The intuition for (3) is that the mutual influence between two firms increases with their size, agreement in action, and sector commonality. We also define function $\phi(ch_i, s_i)$ to compare i's action to its change parameter ch_i. Specifically, ϕ's value is positive if s_i is upgrade (retain) and ch_i is greater (smaller) than 0.5. The interpretation is that a value greater (smaller) than 0.5 for ch_i implies that i is flexible (inflexible) with respect to technology change.

$$\phi(ch_i, s_i) = \begin{cases} \frac{s_i}{2} - ch_i & \text{if } \frac{s_i}{2} - ch_i < 0.5 \\ 0.5 - \frac{s_i}{2} + ch_i & \text{otherwise} \end{cases}$$

Finally, the overall payoff combines the pairwise partnership weights, further adjusted by the company's flexibility in upgrading its technology.

$$u_i(a_{N_i}) = (1 + y_i \phi(ch_i, s_i)) \sum_j w_{ij}(s_i, s_j),$$

with $y_i \sim U[0, 1]$.

3.2 GMM CONSTRUCTIONS

Given the payoff function and the game definition above, we can generate a regret potential function (2) in a straightforward manner, parametrized by temperature. This potential function in turn defines a regret GMM, *reG*.

Our heuristic rule-based model, *hM*, in contrast, is specified without direct reference to the payoff function. In this model, each company independently applies a local heuristic to stochastically choose its action. Specifically, agent i changes its technology with

probability $pChange(i)$, where

$$pChange(i) = 0.5(1 - 10^{-3})^{|N_i|}(1 - 10^{-3}z_i).$$

The intuition behind this heuristic is that the more partners ($|N_i| - 1$) a company has and the greater its size z_i, the less likely it is to change. Given the *pChange* values, it is straightforward to define a potential function for the GMM hG such that the outcome distribution is the same as generated by applying the rule independently for each company. As a result, hG's potential π_i is a function of only s_i instead of s_{N_i}.

The two GMMs, *reG* and hG, are based on qualitatively different sources of knowledge. If we believe that agents are essentially rational and aware of their environment, we may expect that the regret GMM *reG* would predict behavior well. If instead we have evidence that agents choose heuristically based on partnership density and size, we might have greater confidence in predictions based on hG, or on some other heuristic models that capture our intuition of agents' behavior. In other situations, we may consider that both models capture factors determining behavior, and view the knowledge sources as complementary.

3.3 SIMULATION MODEL

The role of our simulation model is to generate play data from a plausible agent interaction process. In this study, we treat this data as the actual outcome, and use it to evaluate the GMMs above as well as combined models. The simulation is based on the idea that actual agent behavior is produced via repeated interaction through a reinforcement learning (RL) procedure.

In the model, each agent is an independent learner, employing an RL procedure designed for partially observable environments (Jaakkola et al., 1995). The environment for company i comprises i and its partners (i.e., its neighbors N_{-i}), and each configuration of partner strategies $s_{N_{-i}}$ is a possible state. The agent seeks to learn a stochastic play policy: $\sigma_i(s_i \mid s_{N_{-i}})$, which denotes the probability of playing action s_i at state $s_{N_{-i}}$. However, the agent does not actually observe $s_{N_{-i}}$ before taking its action. Thus, the action is actually selected based on the policy, for $a = 1, 2$,

$$\Pr_i(s_i = a) = \sum_{s_{N_{-i}}} \sigma_i(s_i = a \mid s_{N_{-i}}) \Pr(s_{N_{-i}}). \quad (4)$$

$\Pr_i(s_{N_{-i}})$ is a stationary probability distribution over states, which, for simplicity, we take to be uniform. The scarcity of companies' knowledge about others' strategies and the network effects motivates our choice of $\Pr_i(s_{N_{-i}})$ instead of a more complicated model of network dynamics.

To learn the policy σ_i, we apply the RL procedure of Jaakkola et al. (1995).

1. Initialize σ_i to $pChange(i)$ (i.e., the heuristic policy from hM).
2. Generate a play s using (4). Observe the resulting local state s_{N_i} and receive as reward the payoff $u_i(s_{N_i})$. Update $Q_i(s_i, s_{N_{-i}})$, the average reward for taking action s_i in the associated local state.
3. Choose $\sigma_i^*(s_i \mid s_{N_{-i}})$ to maximize $Q_i(s_i, s_{N_{-i}})$. Adjust the current policy σ_i in the direction of σ_i^*: $\sigma_i \leftarrow \sigma_i(1-\gamma) + \sigma_i^*\gamma$, where γ is the learning rate.
4. Repeat steps 2 and 3 until convergence.

For our experiments, we used $\gamma = 0.2$ and iterated steps 2 and 3 above 40 times, the point after which few changes occurred in the learned RL policy σ. We denote the simulated model at the end of the RL procedure as *simM*.

Note that the RL process starts with the local heuristic rule-based policy, but is updated based on payoff experience. Thus we expect that both the heuristic rule-based model and the rationalistic regret-based model may offer value for predicting the outcomes of *simM*.

4 METHODS FOR COMBINING KNOWLEDGE SOURCES

Given two complementary sources of knowledge, how can we integrate them into a single GMM? We formulate the problem for the case that one knowledge source is expressed explicitly as a GMM, G_1, and the other in the form of some data $D = \{s^1, \ldots, s^m\}$ of joint plays related to the multiagent scenario.[2] Note that D may not reflect the actual distribution of play accurately, for example because it is small in size or because it was observed in a different multiagent setting. In this section we answer these questions abstractly and in the next section show how we can do this for our specific *reG* model and data D derived from hM.

4.1 DIRECT UPDATE

The *direct update* method combines the two sources of knowledge G_1 and D into a new GMM, *directG*, derived by adjusting the λ_{G_1} parameters of G_1 to maximize the predictive performance w.r.t. the data D.

[2]Since we can generate such a data set from a GMM, or induce a GMM from data, the combination methods can be applied to more general settings where knowledge sources come in either form.

We measure predictive performance using the logarithmic scoring rule (Gneiting and Raftery, 2007): $\text{Score}(G \mid D) = \sum_{k=1}^{|D|} \log \Pr_G(s^k)$, which assesses the log-likelihood of the data. We take as our problem to tune the GMM's parameters λ in order to maximize this score (Kappen and Rodríguez, 1997),

$$\text{Score}(G_1 \mid D) = \sum_{k=1}^{|D|} \log \Pr_{G_1}(s^k) = L(D \mid \lambda = \lambda_{G_1}).$$

We employ the gradient ascent method to maximize data likelihood, which entails computing the gradient of L w.r.t λ:

$$\begin{aligned} \nabla\lambda &= \frac{\partial L(D|\lambda)}{\partial \lambda} \\ &= \frac{\sum_k^{|D|} \partial \log e^{-\sum_i \lambda_i \epsilon_i(s^k_{N_i})}}{\partial \lambda} - |D| \frac{\partial \log Z}{\partial \lambda} \end{aligned} \quad (5)$$

and adjusting $\lambda = \lambda + \alpha\nabla\lambda$, where α is the learning rate, until the gradient is below some threshold.

A major problem in graphical-model parameter learning is the intractability of calculating $\log Z$ in (5). It entails iterating over all possible outcomes of the game, which is exponential in the number of players N, rendering exact inference and learning in undirected graphical models intractable. Since our priority is to accurately evaluate knowledge combination methods for GMMs, our current implementation of the inference and learning algorithms does not employ any approximation. Our pilot study of generalized belief propagation approximations in GMMs (Yedidia et al., 2001) has indeed yielded positive results, and will be incorporated in future reports.

4.2 OPINION POOL

Unlike direct update, which depends on the availability of both play-outcome data and the potential function's parameterized form, the next two methods, *opinion pool* and *mixing data*, push the knowledge combination problem towards potentially greater independence from the input knowledge sources' forms.

The *opinion pool* method starts by first using half the given data D to learn a GMM G_2 by adjusting its parameters to maximize the likelihood of the data, as in the direct update method above. The combined model OPG is then an aggregation of G_1 and G_2 into a single probability distribution:

$$\Pr_{OPG}(s) = f(\Pr_{G_1}(s), \Pr_{G_2}(s)).$$

Note that the above equation does not involve defining a separate pooled potential function for each player in the combined model. The rationale is that potentials are not normalized like the joint probability, and thus, their absolute values contain little meaning when taken out of the context of their corresponding models.

We adopt for our aggregation function the weighted geometric mean, called the *logarithmic opinion pool* (logOP).

$$\Pr_{OPG}(s) = \frac{\Pr_{G_1}(s)^w \Pr_{G_2}(s)^{1-w}}{Z}.$$

The logOP is the only pool known to preserve independence structure in graphical models (Pennock and Wellman, 2005), which is an important property in our context. The weight parameter w can be set by fiat, or tuned with data. In our experiments described below, we employ the other half of input game-play data D, denoted $\bar{D}$, and set w by maximizing the objective function:

$$L(\bar{D} \mid w) = \sum_k^{|\bar{D}|} \log \Pr_{OPG}(s^k). \quad (6)$$

In brief, given the two components G_1 and G_2, the opinion pool method first initializes $w = 0.5$. It then repeats computing the gradient ∇w of the log-likelihood w.r.t w by differentiating (6), and updating $w = w + \beta\nabla w$, where β is the learning rate, until the gradient is acceptably small.

4.3 MIXING DATA

The *mixing data* method samples joint plays from the given GMM G_1 to generate a new data set D_1. We combine D_1 and D into one data set mD by sampling from the two sources equally, though one could easily weight one source more than another by adjusting the sampling ratio. We then induce a new GMM for a given parametrized form by tuning the parameter λ, as detailed in the direct update approach (Section 4.1), to maximize the likelihood of the new data mD.

Below is the outline of the mixing data method, which produces the combined GMM *mixG*. Note that we leave out step 4 in our implementation.

1. Generate a sample of play outcomes D_1 from G_1.
2. Build the mixed data set mD from D_1 and D with sampling ratio $\omega = 0.5$.
3. Initialize *mixG* with some λ_{mixG}. Update λ_{mixG} as in direct update using the data mD.
4. (optional) Tune ω in the direction determined by the gradient-descent method to maximize *mixG*'s performance on a small held-out part of the testing data. Repeat steps 3 and 4 until the gradient is below some threshold.

5 EMPIRICAL STUDY

We evaluate our combination approaches experimentally using our simplified version of the Internet industry partnership domain. We concentrate mainly on Example 1, the scenario depicted in Figure 1. In the first experiment, we also examine Example 2, which employs a smaller graph including only the top four companies.

5.1 EXPERIMENT SETTINGS

In our experiments we use the following components defined above: (i) the regret GMM *reG* with the temperature parameters generated uniformly randomly (except in one case explicitly specified below), (ii) a data set D of joint plays generated by using the heuristic rule-based model hM, (iii) the heuristic model hM and associated GMM hG, and (iv) a testing data set D^* derived from the RL-based model $simM$. We compare our different combination approaches based on their ability to predict the test data set as measure by the score function $\text{Score}(G \mid D^*)$. In particular, we compare the performance of a model that combines knowledge sources *combinedG* relative to the performance of a baseline model *baseG* using a ratio of scores, $R = \frac{\text{Score}(baseG|D^*)}{\text{Score}(combinedG|D^*)}$.

The inverted order of *baseG* and *combinedG* is due to Score's negativity, and thus, any $R > 1$ indicates *combinedG*'s improvement over *baseG*.

We experiment with several environment settings. For each setting, we conduct 20 trials, each of which involves a training data D set of 500 plays from hM and a testing data set D^* of 500 plays from $simM$.

5.2 RESULTS AND ANALYSIS

First, in Figure 2 and Figure 3 we present an overview of our combination methods' effectiveness. For both figures, *reG* and D are used to derive the model *directG* using the direct update combination method, the model *OPG* using the opinion pool method, and the model *mixG* using the mixed data method.

Figure 2 displays predictive performance across the two examples (1 and 2) and the two baseline models (*reG* and *hG*). Mixing data is consistently best of the three combination methods, and direct update performs relatively better than opinion pool. All three methods yield better results than individual input models, suggesting that combining two knowledge sources is beneficial regardless of which of the proposed methods is adopted.

Figure 3 shows the performance of various models compared to a model derived from the same gold-standard source *simM* as our test data D^*. We sample a separate data set D' from *simM* and employ it in learning a GMM *simG* of the parametrized regret form (2), using maximum likelihood to adjust the temperature parameter. The results reveal that our combined models, especially *mixG*, closely match *simG* in terms of predictive performance.

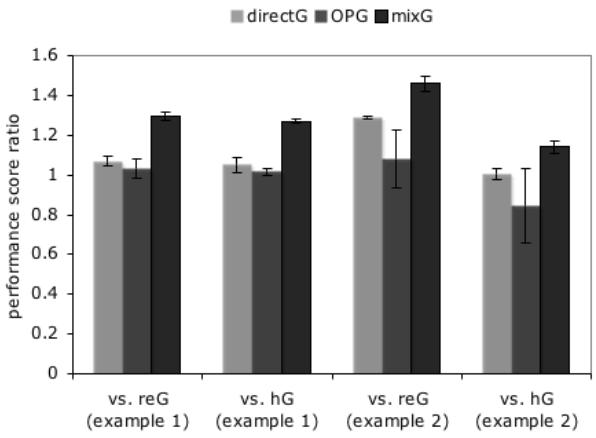


Figure 2: Combination methods' performance across different examples and baselines.

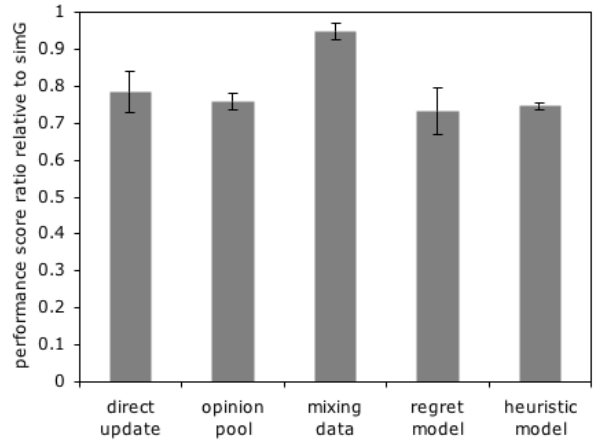


Figure 3: Combination and input models versus the underlying model.

Next, we study the effect of varying the quality of the two input sources. Figure 4 shows the effect of varying the amount of joint-play data available in D. Specifically, we make a fraction $\rho|D|$ of play observations, $\rho \in [0, 1]$, available to the three combination methods. From Figure 4, we observe that as long as $\rho > 0.1$, performance remains fairly stable. In our experiments, this corresponds to a threshold data set size of approximately $500 \times 0.1 = 50$. When the amount of data goes

below this threshold, the combined model may very well become inferior to the models reG and hG.

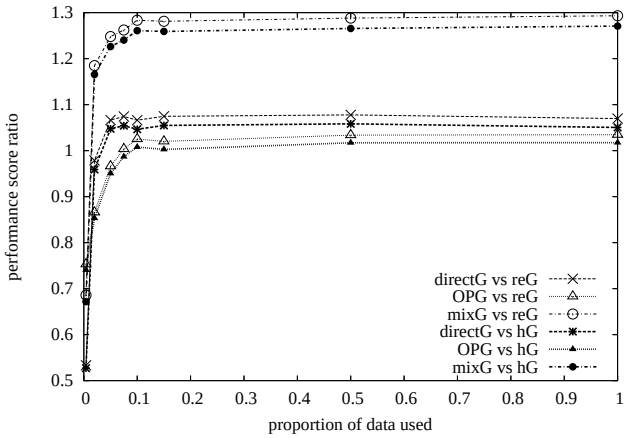


Figure 4: Combination methods' performance versus data availability portion ρ.

Figure 5 shows the effect of varying the quality of the GMM provided as input to the combination methods. To modulate the accuracy of reG, we introduce a parameter δ controlling the relation of its temperature parameters to those of $simG$. Specifically, we set λ_{reG} to $(1+\delta)\lambda_{simG}$. The results of the third experiment are depicted in Figure 5. When compared with the unchanged heuristic model hG, the combination models show a slight decrease in their relative performance with δ, which reflects the effect of reG's inaccuracy on the combination methods. When the baseline is reG, in contrast, the degradation of the combined model is dominated by the effect of compromising the baseline reG. In other words, combining knowledge sources effectively compensates for degrading one of them.

In these experiments, OPG's poor performance relative to that of $directG$ and $mixG$ may be due in part to its reliance on only a single parameter, w, compared to the vector λ available to the other methods. $mixG$'s overall superiority is likely a result of its directly sampling from reG, which employs information contained in reG more effectively than $directG$, where reG only matters at the initialization stage.

Figure 6 presents the results of an experiment designed to strengthen our claims about the benefits of integrating knowledge sources in a single model. We examine the combined models' performance in environments where the simulation mode $simM$ is not the product of RL that starts with the input model hM. In particular, we define a different heuristic model hM', such that $pChange_{hM'}(i) = 0.05$ for all i. Let E be the input data set generated from hM'. Based on hM',

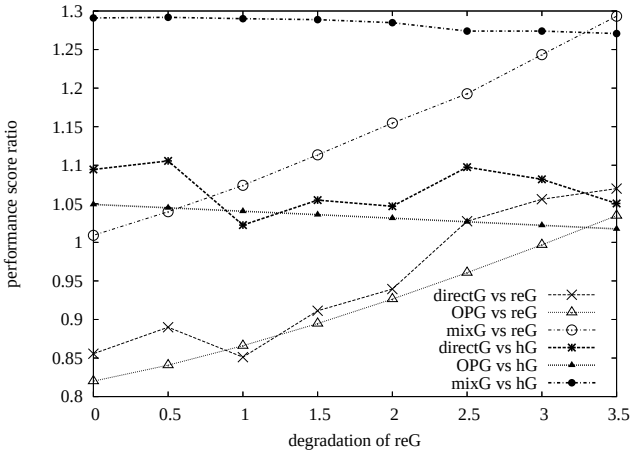


Figure 5: Combination models' performance versus input regret model's inaccuracy as controlled by δ.

we subsequently build the simulation model $simM'$ and its corresponding test data E^*. Given these data sets and models, we can evaluate the combined models across drastically different starting points, by comparing their performances when different input sources, D and E, are provided, on the same testing data (either D^* or E^*). First, we compare the performance of the heuristic models employed in generating input data, $hG_{(D)}$ (induced from hM) and $hG_{(E)}$ (induced from hM'): $hG_{(D)}$ performs 60% better than $hG_{(E)}$ when tested on D^*, whereas $hG_{(E)}$ outperforms $hG_{(D)}$ by 54% on E^*. This assessment affirms that the two different input data sets, D and E, are indeed differentiable in terms of the behavior models they represent.

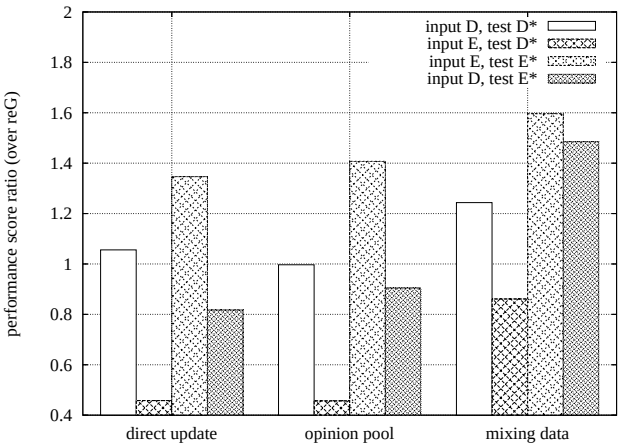


Figure 6: Combination methods in different input and test settings.

The results presented in Figure 6 indicate that better

performance is achieved in cases where the test and input environments coincide (the first and third measures) compared to those in which they are unrelated (second and fourth). This observation confirms that combined models whose input data contains some information about the underlying behavior model outperform those extracting irrelevant information from its inputs. The gaps in $mixG$'s performance between scenarios where test cases are derived from the same input (first and third) and from an unrelated model (second and forth) are relatively smaller than those in the other methods. This phenomenon is likely a result of $mixG$'s more effective usage of both knowledge sources, which limits the impact of irrelevant input data, and possibly contributes to the good performance of $mixG_{(D)}$, which is tuned to fit D, when tested on E^*.

6 CONCLUSIONS

GMMs provide a flexible representation framework for graphically structured multiagent scenarios, supporting the specification of probability distributions based on game-theoretic models as well as heuristic or other qualitatively different characterizations of agent behavior. We explored the possibility of exploiting this flexibility by employing multiple knowledge sources for prediction, as demonstrated for the task of predicting the outcome of a reinforcement learning process.

Our basic finding is that combining two knowledge sources in this scenario does improve predictive power over either input source alone. We have also identified the most effective combination method among those tried—mixing data—and the existence of a threshold for data availability that can help boost efficiency. Furthermore, we have found that our knowledge combination approaches, especially mixing data, can effectively match the performance of modeling the reinforcement learning process directly.

This study is a first step in what we expect to be an extended effort to develop the GMM framework for supporting reasoning about strategic situations. One important issue to address is computational feasibility; although the graphical representation facilitates scalability in the number of agents, accurate approximation techniques are nevertheless essential to support practical applications with large models.

Knowledge about multiagent behavior may come from sources other than play history and regret functions, and so another logical research direction is to develop canonical models for capturing and combining such sources. Learning may be extended to not only the model's parameters, but also the structure of potential functions and the graph topology itself. Such extensions present more complicated problems for combining models derived from different knowledge sources. Finally, we might look to dynamic Bayesian network concepts to extend the GMM framework to add a time dimension, and thus enable modeling sequential and interactive multiagent environments.

References

Daskalakis, C. and Papadimitriou, C. H. (2006). Computing pure Nash equilibria in graphical games via Markov random fields. In *Seventh ACM conference on Electronic Commerce*, pages 91–99, Ann Arbor.

Gal, Y. and Pfeffer, A. (2008). Networks of influence diagrams: A formalism for reasoning about agents' decision-making processes. *Journal of Artificial Intelligence Research*.

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction and estimation. *Journal of the American Statistical Association*, 102(477):359–378.

Jaakkola, T., Singh, S., and Jordan, M. I. (1995). Reinforcement learning algorithm for partially observable Markov decision problems. In *Advances in Neural Information Processing Systems 7*, pages 345–352.

Kappen, H. J. and Rodríguez, F. B. (1997). Mean field approach to learning in Boltzmann machines. *Pattern Recognition Letters*, 18:1317–1322.

Kearns, M. (2002). Computational game theory: A tutorial. `http://www.cis.upenn.edu/~mkearns/nips02tutorial/nips.pdf`. Presented at the *Neural Information Processing Systems* conference.

Kearns, M. (2007). Graphical games. In Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V., editors, *Algorithmic Game Theory*, pages 159–180. Cambridge University Press.

Kearns, M., Littman, M. L., and Singh, S. (2001). Graphical models for game theory. In *Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 253–260, Seattle.

Koller, D. and Milch, B. (2003). Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45:181–221.

Krebs, V. (2002). Internet industry partnerships. `http://www.orgnet.com/netindustry.html`.

Pennock, D. M. and Wellman, M. P. (2005). Graphical models for groups: Belief aggregation and risk sharing. *Decision Analysis*, 2:148–164.

Vorobeychik, Y. and Wellman, M. P. (2006). Mechanism design based on beliefs about responsive play (position paper). In *ACM EC-06 Workshop on Alternative Solution Concepts for Mechanism Design*, Ann Arbor, MI.

Wolpert, D. H. (2006). Information theory: The bridge connecting bounded rational game theory and statistical physics. In *Complex Engineered Systems*. Springer.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2001). Generalized belief propagation. *Advances in Neural Information Processing Systems*, 13:689–695.

Almost Optimal Intervention Sets for Causal Discovery

Frederick Eberhardt*
Institute of Cognitive and Brain Sciences
University of California, Berkeley
Berkeley, CA 94720; and

Abstract

We conjecture that the worst case number of experiments necessary and sufficient to discover a causal graph uniquely given its observational Markov equivalence class can be specified as a function of the largest clique in the Markov equivalence class. We provide an algorithm that computes intervention sets that we believe are optimal for the above task. The algorithm builds on insights gained from the worst case analysis in Eberhardt et al. (2005) for sequences of experiments when all possible directed acyclic graphs over N variables are considered. A simulation suggests that our conjecture is correct. We also show that a generalization of our conjecture to other classes of possible graph hypotheses cannot be given easily, and in what sense the algorithm is then no longer optimal.

1 INTRODUCTION

Suppose we have a set of variables $\mathbf{V} = \{X_1, \ldots, X_n\}$ and we are interested in discovering the causal relations among these variables. That is, we want to find out which variable influences which other variable directly or indirectly. This problem is common in both the natural sciences (e.g. gene regulatory networks, drug testing etc.) and the social sciences (e.g. econometrics, policy decisions etc). Depending on the area of research the scientist will have different tools to investigate this problem and different model space assumptions and background knowledge will be appropriate or available. The main problem for the scientist is to determine and justify these assumptions and to use tools of investigation that will distinguish the true causal structure from other possible causal structures, i.e. the scientist must reduce and, if possible, rule out underdetermination of the causal structure.

For any particular set of causal variables, the scientist will consider a set of possible hypotheses that describe the causal relations among these variables. Causal Bayes nets (Spirtes et al. 2000; Pearl 2000) provide a concise framework by representing causal structures in terms of directed acyclic graphs (DAGs) connecting the variables, and a probability distribution over these variables, that factors according to the DAG. If nothing is known about the causal structure then the set of hypotheses is large, namely, all possible DAGs over the set of variables, including maybe structures that involve latent variables. For these cases search procedures can and have been devised (see Eberhardt 2007 for details and references) that – given a set of assumptions over the model space – determine the causal structure uniquely, or give a precise specification of the remaining underdetermination. In particular, the following result is relevant to this paper. In Eberhardt et al. (2005) we showed that:

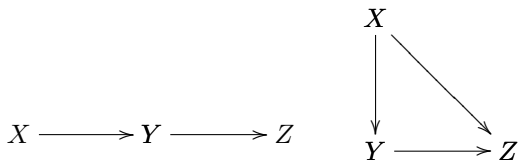
Theorem 1.1 (Worst Case Bound for Complete Hypothesis Space)**.** *Given a set of N causally sufficient variables $\lfloor \log_2(N) \rfloor + 1$ experiments are sufficient and in the worst case necessary to discover the causal structure uniquely if multiple variables can be subject to an intervention simultaneously and independently.*

That is, if all that is known about the true causal structure among N variables is that there are no latent common causes, then the structure can be determined uniquely in at most $\lfloor \log_2(N) \rfloor + 1$ experiments, where each experiment may involve a randomization of several variables. The bound is based on the combinatorics of different ideal experiments, assuming one has access to an oracle for the independence constraints of the (manipulated) distribution, i.e. it applies to the large sample limit.

But in actual science, it is rarely the case that the

*Also: Department of Philosophy, Washington University in St. Louis, St. Louis, MO 63130; Contact: fde@berkeley.edu.

hypothesis set is so general and uninformative. Often substantial knowledge about parts of the causal structure is already available, and only a search among a restricted set of hypotheses is required. In this case the search procedures implied by the worst case bound are overkill. Something more adaptive is required. Take a simple case, where we only have three variables X, Y and Z and suppose we are sure (for whatever reason) that they can only be causally related as in one of the following two DAGs:



Suppose, for the sake of argument, that no other variables – measured or unmeasured – are relevant, i.e. X, Y and Z are causally sufficient. In this case, the search procedure is simple: The cheapest way of distinguishing the DAGs is probably to collect passive observational data: If $X \perp\!\!\!\perp Z | Y$ then the first structure is true, otherwise the second. But as is well known, passive observation does not distinguish all possible causal structures. In particular, the following three structures cannot be distinguished based alone on the independence relations they imply:

$$X \to Y \to Z \quad X \leftarrow Y \to Z \quad X \leftarrow Y \leftarrow Z$$

Together they form a so-called observational Markov equivalence class (OME) of causal structures, where each DAG in the OME implies the same independence constraints. There are more and less obvious ways to proceed: A less obvious one is that if the causal relations among the variables form a linear structural equation model with *non*-Gaussian error terms, then these three causal hypotheses can be distinguished with passive observational data alone (Shimizu et al. 2006). The more obvious procedure is to perform an experiment: If we intervene with a randomization on Y then we are able to uniquely distinguish the three hypotheses: If the first one is true, then under the randomization of Y we have $X \perp\!\!\!\perp Y$, since the randomization makes the intervened variable (Y) independent of its normal causes (X in this case). This feature of randomization is what is sometimes referred to as "edge-breaking". We do not find this independence for the other two structures under the same manipulation. For similar reasons, if the third hypothesis is true, then $Y \perp\!\!\!\perp Z$ under a randomization of Y, whereas under the same circumstances $Y \not\perp\!\!\!\perp Z$ for hypotheses one and two.

Obviously, if the hypothesis set were different, then an experimental intervention on Y would not necessarily distinguish (all) the hypotheses. In the simplest case, suppose we just have two variables and just two possible hypotheses: Either X causes Y or the two variables are causally separate. In that case an intervention on Y would be useless, since under a randomization of Y the two structures are equivalent: Both imply that $X \perp\!\!\!\perp Y$ given that Y was randomized. Depending on the hypothesis space and the sequence of experiments, underdetermination can be difficult to resolve. So, analogously to an OME, we refer to structures that imply the same independence constraints in a sequence of experiments as forming a manipulated Markov equivalence class (MME) for that sequence of experiments.

Given OMEs and MMEs as common examples of more restricted hypothesis spaces, the obvious question is whether we can provide efficient search procedures that ensure that we reduce the underdetermination as fast as possible. Can we state guarantees on those search procedures, just as Theorem **??** gives guarantees for search procedures on the complete hypothesis space?

1.1 BACKGROUND

Traditionally, causal discovery algorithms split into two categories: constraint based and score based algorithms. Score based algorithms, such as the GES-algorithm (Chickering 2002), compute a score for each model given the measured data. The model with the highest score is then deemed to be the most likely or most plausible model given the data. In contrast, constraint based algorithms (e.g. the PC-algorithm (Spirtes et al. 2000)) test for particular constraints in the measured data and select models that match the discovered constraints. Most commonly, independence constraints are used, but many other constraints can also be considered.

Without further assumptions (such as time order, non-Gaussian errors etc.) both types of search algorithms are asymptotically limited to the discovery of OMEs or MMEs of causal structures. But as structure search algorithms they are uninformative about which experiment(s) should be performed, if a sequence of experiments is necessary to further reduce underdetermination. Tong & Koller (2001), Murphy (2001) and Cooper and Yoo (1999) have presented Bayesian approaches on how to select the next experiment *given* an OME or MME. Generally speaking, all of their approaches are based on selecting the next experiment that maximally reduces the entropy among the remaining hypotheses. This is computationally extremely expensive since an optimization over all possible parameterizations of all possible models has to take place. All of the cited approaches use different computational shortcuts to approximate the computation.

A similar procedure based on the qualitative graphical structure, which would be much more suited for constraint based algorithms, has not been given, although various suggestions for measures of selecting the next best experiment in such circumstances have been made in Meganck et al. (2005). We are unaware of any algorithmic implementations of how to compute the intervention sets efficiently given the measures, or of bounds on the search procedure that these measures imply. The advantage of an algorithm for the selection of experiments based only on qualitative features of the causal structure is that it does not require any distribution over the hypothesis space or any parameterization of the causal structures, both of which may only be rarely available at the beginning of actual scientific inquiries.

1.2 CLASSES OF HYPOTHESES

There are many ways in which information may be available that restricts the hypothesis set of causal structures. We will use a *knowledge graph* to characterize types of structural constraints that restrict the hypothesis set.

Definition 1.2 (Knowledge Graph)**.** *A knowledge graph is a mixed graph over a set of vertices* $\mathbf{V}$ *such that any two vertices are connected by exactly one of the following edge-types:*

direct cause: *A directed edge represents the knowledge that one vertex, the start, is a direct cause of the other, the end (relative to* $\mathbf{V}$*):* $X \longrightarrow Y$

non-adjacency: *The absence of an edge represents the knowledge that neither vertex is a direct cause of the other:* $X \quad Y$

adjacency: *An undirected edge represents the knowledge that there is a direct causal connection between the two vertices, but that it is not known in which direction it goes:* $X \longrightarrow\!\!\!\!\!- \; Y$

semi-directed: *A semi-directed edge from vertex* X *to* Y *represents the knowledge that neither variable is a direct cause of the other or that* X *is a direct cause of* Y*:* $X \dashrightarrow Y$

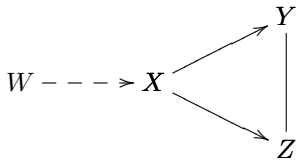
no knowledge: *A no-knowledge edge represents the lack of any knowledge about the direct connection between the two vertices:* $X -\!\!?\!\!- Y$

An edge in a knowledge graph is considered known *if it is of one of the first two edge-types, otherwise it is unknown. A knowledge graph is said to represent a* causal structure *uniquely when each of its edges is* known *and the structure is acyclic.*

A knowledge graph, like a pattern for an OME, can be used to represent an equivalence class of graphs that imply the same independence constraints. The main difference to a pattern is that a knowledge graph can represent information about independence relations resulting from interventions. Clearly, an OME of a causally sufficient set of variables can be represented as a knowledge graph, since only the first three edge-types are required. The fourth edge-type is used when exactly one variable of a pair is subject to an intervention and the pair is found to be independent in the experimental data set. The fifth edge-type is used when no information is available about the causal relation between two variables, for example when two variables are subject to interventions simultaneously. Knowledge graphs can represent OMEs and MMEs, but not all knowledge graphs represent classes of graphs that are OMEs or MMEs. For example, for three variables, if each pair is connected by a semi-directed edge, then we would have a knowledge graph, but no passive observation or sequence of one or more experiments would ever yield such an equivalence class.

1.3 EXAMPLE

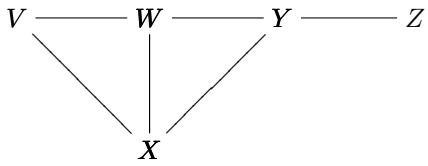
We can now illustrate with an example how intervention sets should be selected when the hypothesis space is restricted. Suppose that for a set of four variables W, X, Y, Z, the following knowledge graph is known – it could be the output of a causal search algorithm given data generated in an experiment where X was randomized:



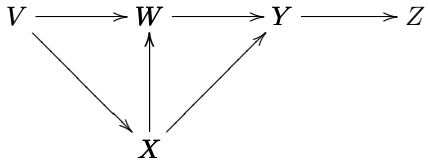
Clearly, the next intervention should be an intervention on Y or Z (and possibly, but not necessarily W also). To resolve the orientation of the undirected edge we must intervene on one of its endpoints: independence between the endpoints will indicate an incoming edge on the intervened endpoint (given the knowledge graph), while dependence for all conditioning sets will indicate an edge outgoing from the intervened endpoint. The semi-directed edge can be resolved if both its endpoints are passively observed or if W is subject to an intervention. In both cases, if W and X are found to be dependent, then – given the knowledge graph – we know that $W \rightarrow X$, otherwise we know that there is no direct causal link between the variables.

This was easy. Now suppose we have an OME over the set of variables $\mathbf{V} = \{V, W, X, Y, Z\}$ represented

by the following knowledge graph:[1]



This knowledge graph represents 12 distinct causal hypotheses (substantially less than the 29,281 possible DAGs over 5 variables). Without further assumptions, passive observational data is not going to help. But we can intervene, the question is where? Note that if we can orient one edge, we can determine additional orientations that are implied (see Meek Rules in Meek 1995). For example, if we find that $Z \to Y$, then this also implies $Y \to W \to V$ and $Y \to X \to V$. But intuitively, intervening on Z might seem like a bad idea given this OME. X or W may appear to be better candidates, and rightly so: The advantage of intervening on X is that there are particular structures (two in fact) in the equivalence class, that, if true, would be uniquely distinguished within the equivalence class by a single experiment randomizing X. The following is one such structure:

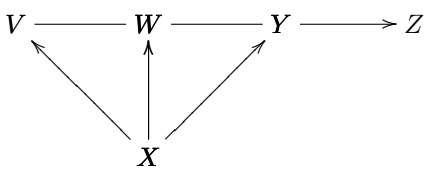


The same is true, albeit for other structures, for interventions on V, W and Y, but not for an intervention on Z: No matter what the true structure is, an intervention on Z will not uniquely resolve this Markov equivalence class. Further, brief inspection shows that there is no intervention set (of any size) that *guarantees* discovery of the true causal structure in one experiment.

How does one compute intervention sets for knowledge graphs for which intuitions fail us?

The algorithm we propose builds on insights from the analysis underlying Theorem **??**. The key difficulty in uniquely identifying the true causal structure is to determine the orientation of edges in cliques. Cliques are subsets of the vertex set for which every pair of vertices is connected by an edge in the true causal structure. A clique makes edge-orientations maximally independent, because fewer orientations are implied (there are no v-structures[2]; only acyclicity constraints imply orientations). An algorithm aiming to minimize the number of experiments must therefore break down cliques of connected variables as fast as possible. For a clique of size $|C|$, an experiment that intervenes on k variables in the clique determines the orientation of $k(|C| - k)$ of the clique's edges. This value is maximized for $k = |C|/2$. So our algorithm, presented below, works as follows:

Given the knowledge graph it determines all maximal cliques of variables connected by so-called *unknown* (undirected, semi-directed or no-knowledge) edges. In our example we have: $C_1 = \{V, W, X\}, C_2 = \{W, X, Y\}$ and $C_3 = \{Y, Z\}$. However, we only have to consider C_1 and C_2, since orientation of all edges in a 3-clique will take 2 experiments in the worst case, and so we can leave resolution of C_3 to the second experiment, since it only requires a single experiment (and may anyway get resolved for free in the first experiment). Ideally we should intervene on $|C_i|/2$ variables simultaneously for each clique C_i. However, since the cliques may (and in our case do) overlap, the selection of the variables to subject to interventions cannot be done for each clique independently. Consequently, we count for each vertex how many cliques it is part of to determine which vertices are in the most cliques. By considering maximal cliques in order of size and their vertices in order of their prevalence in the cliques, the algorithm greedily approximates the optimal choice of intervention set. In our example, W and X are in 2 (relevant) cliques, while V and Y are only in one. The choice between X and W is random, since there is no a priori reason to prefer one over the other. Once, say, X is selected, all cliques are updated as to whether a sufficient number of their vertices are contained in the intervention set (see algorithm for details). In our case, the selection of X for the intervention set would imply that C_1 and C_2 have sufficient vertices in the intervention set, since inclusion of any additional vertex from those cliques would not necessarily reduce the number of experiments needed. (Note, that one could include Z in the intervention set to resolve C_3, but one need not.) Suppose that upon intervening on X we find the same independence constraints as in the OME. We can then update our knowledge graph to reflect the intervention test:



We were unlucky; a further experiment is required, but note that the orientation of the YZ-edge was implied by the orientation of the XY-edge resulting from the intervention. Now only the V–W–Y connection has

[1] We use the example from Meganck et al. (2005).

[2] Variables X, Y, Z form a v-structure if $X \to Y$ and $Z \to Y$ and there is no direct causal connection between X and Z. V-structures are also known as unshielded colliders, and due to their unique independence implications, can be identified in passive observational data.

to be resolved: W is added to the new intervention set for the next experiment, since it is the only vertex that is part of two cliques. We know from the discussion earlier that this intervention will be sufficient to uniquely determine the true graph. Our algorithm determined intervention sets that could have resolved the OME in a single experiment, but that ensured that it was resolved in what we knew to be the worst case theoretical bound for this particular knowledge graph: two experiments.

2 THEORY

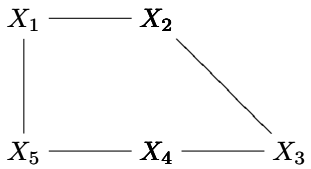
We conjecture that the above procedure works quite generally for knowledge graphs that are OMEs, and that the number of experiments that are in the worst case necessary, is a function of the largest clique in the OME:

Conjecture 2.1 (Experiments on OMEs)**.** *Given an OME of the true graph, if multiple simultaneous and independent interventions can be performed in each experiment, then* $\lceil \log_2(|C_{\max}|) \rceil$ *experiments are sufficient and in the worst case necessary to recover the true causal graph, where* $C_{\max}$ *is the largest clique in the OME.*

Depending on the size of the largest clique in the OME, this bound is substantially lower than the bound of Theorem **??**. But if the largest clique in the OME contains all N variables, then the bounds coincide. (The one additional experiment in Theorem **??** only makes a difference when N is a power of 2. In those cases the adjacency information – present in an OME, but not assumed in Theorem **??** – cannot be established completely in the $\log_2(N)$ experiments.)

An algorithm that reduces by half the size of all cliques of undirected edges, for which no edge-orientation is known, clearly satisfies the conjectured bound. In general, the requirement is a little weaker: If all undirected cliques $C_{>h}$, that are larger than h variables, where h is the closest power of 2 below $|C_{\max}|$, are reduced to cliques of size h in each experiment, then the conjectured bound is satisfied.

The conjecture currently remains without proof because it is not entirely clear whether it is possible to find intervention sets for any OME that break down all such cliques sufficiently fast. Since cliques may overlap, a proof of the conjecture must guarantee that intervention sets can always be found that resolve all of the overlapping cliques at once. In particular, consider a knowledge graph containing an undirected five-cycle as shown below. Here we have four overlapping maximal cliques of size 2. Again, one experiment should be sufficient to recover the causal structure, but in this case it is impossible to find an intervention set that resolves all four cliques simultaneously.



Any selection would result in one clique for which either both or no variable is contained in the intervention set. However, the five-cycle is not a counterexample to the conjecture since an undirected five-cycle (or any other cordless cycle greater than three) cannot occur in any OME – it would always imply a v-structure). Hence, the conjecture does not hold for arbitrary knowledge graphs, but is dependent on the assumption that the graph is an OME or derived from one.

For arbitrary knowledge graphs there is a general negative graph theoretic result due to Folkman (1970). It relates the problem of intervention set selection to coloring theorems:

Theorem 2.2 (Folkman Clique Theorem – paraphrased)**.** *For any clique-size* $c \geq 3$*, there is a graph* G*, whose largest clique has size* c *and for which every edge two-coloring has a clique of size* c *in one color.*

Considering only the undirected edges of an OME, let an edge be colored red in experiment $\mathcal{E}$ if it connects an intervened and a non-intervened variable, and blue if it connects variables that are both subject to an intervention or both passively observed. In some cases, only the red edges will be oriented as a result of the experiment. Folkman's theorem implies that for any integer $c \geq 3$, there is a graph G whose largest clique has c members. Any coloring, so in particular the coloring we defined, would result in a clique of size c of blue or red edges only. Due to the way our coloring is defined, it is impossible for the clique to be among red edges, since three variables cannot be fully connected by red edges. Consequently, the clique is among blue edges, and since the intervened variables are separated from the non-intervened ones (by red edges), the clique must be either among the intervened variables only or among the unintervened variables only. That is, after the experiment, we may be left with a clique the same size as we started off with. Since there is no way to reduce cliques of un*known* edges by more than half, the conjectured bound does not hold for general knowledge graphs, not even for general adjacency graphs. However, our simulations give hope – as with the five-cycle – that the knowledge graphs that satisfy Folkman's theorem are not OMEs and not derivable from OMEs by sequences of experiments. If that fails, an argument

is needed that such graphs are sufficiently rare so as not to be of practical worry.

3 ALGORITHM

The algorithm takes as input a knowledge graph over the set of variables **V**. It associates with each vertex a boolean field that specifies whether the vertex is admissible to the intervention set or not. A vertex may not be admissible to an intervention set if it is part of a clique for which too many vertices are already part of the intervention set. The free parameter *maxInter* limits how many nodes may be simultaneously subject to an intervention in one experiment. As it stands, the algorithm is not optimal if $maxInter \leq N/2$, but see the discussion below.

Algorithm 3.1 (OPTINTER: Intervention Set Selection)**.** *Given a knowledge graph over a set of vertices* **V***, each vertex in* **V** *can be determined to be* admissible *or* inadmissible *and each vertex has a counter (of clique memberships). Let* maxInter *be the maximum size of the intervention set* **I** *for the next experiment.*

1. *Mark all vertices as* admissible *and set the counters for each vertex to 0.*
2. *Initialize the intervention set* **I** *to be the empty set.*
3. *Find all maximal cliques of vertices connected by un*known *edges and order them* $C_{\max}$ *to* $C_{\min}$ *by the number of vertices they contain. (No need to resolve ties.)*
4. *Each clique can be either* resolved *or un*resolved*. Mark all maximal cliques as un*resolved*.*
5. *Compute* $h = 2^{\lceil \log_2(|C_{\max}|)\rceil - 1}$ *(the closest power of 2 with* $2^n < |C_{\max}|$*).*
6. *Let the* relevant *cliques* $C_1, ..., C_k$ *be the cliques with* $|C_i| > h$*.*
7. *Sort all relevant cliques in order of size, place among equal sized cliques the ones with the most* inadmissible *nodes first.*
8. *Let* C_{curr} *be the first (largest)* unresolved *clique in the list of relevant cliques.*
9. *For each vertex* $U \in C_{curr}$*, set its counter to the number of* unresolved relevant *cliques* C_i *it is part of.*
10. *While* $(|\mathbf{I}| < maxInter) \&\& (|C_{curr} \cap \mathbf{I}| < |C_{curr}| - h)$*, select vertex* $V \in C_{curr}$ *such that* V *is* admissible *and has the highest count; select randomly among ties. Place it in* **I***.*
    (a) *For any relevant clique* C_i*, if* $|C_i \cap \mathbf{I}| = |C_i| - h$*, then mark* C_i *as* resolved*.*
    (b) *For any relevant clique* C_i*, if* $|C_i \cap \mathbf{I}| = h$*, mark its vertices as* inadmissible*.*
11. *Return to 7 and start over until all relevant cliques are* resolved *or when no further relevant cliques can be resolved.*
12. *(Post Process: While possible with regard to the constraints (a) and (b) of step 10, add vertices to the intervention set to resolve additional maximal cliques.)*
13. *Return the intervention set.*

4 SIMULATION

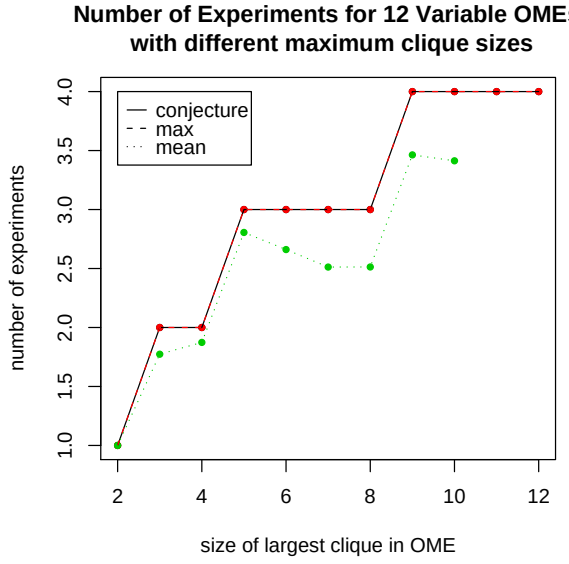
Since we know that Conjecture **??** is not true for general knowledge graphs and we have reason to believe that it holds true for OMEs, we tested the conjecture in a simulation. Since the space of DAGs grows super-exponentially in the number N of variables we have for even very small N a space for which there is no obvious technique on how to generate a reasonable sample of graphs that could be called a representative test of the conjecture. In addition, if we just sampled randomly from the space of all possible DAGs over a set of variables, we would get a sample of graphs which – for the most part – would have very similar (and in fact quite small) maximum clique sizes. Using an MCMC technique to wait for a random DAG with a large clique and then test that, is not feasible given how rare graphs with large cliques are. Constructing graphs with large cliques deterministically runs the risk that our construction method may exclude those graphs that would be a counterexample to the conjecture.

So here is what we did: We considered DAGs with 12 vertices. This was the maximum N for which we could sample and compute a large number of DAGs. We did not consider DAGs with fewer vertices since we assumed that these would occur as subgraphs of the 12-DAGs anyway. By sampling randomly from the space of all possible 12-DAGs we were easily able to generate a large enough number of DAGs whose OME contained maximal cliques with sizes between 1 and 4. To obtain samples of 12-DAGs with OMEs with larger cliques, we constructed 12-DAGs that were completely connected and then randomly deleted 2 edges, and then generated the OME from the remaining DAG. This yielded a decent number of OMEs with cliques between 3 and 10. 11-node cliques are impossible in 12-DAGs if two edges are deleted, but since there is only one DAG – modulo variable renaming – with a maximum clique containing 11 nodes, we confirmed the conjecture in

this case by hand, similarly for the complete 12-DAG. (Consequently they are listed as only "1*" sample in the table below). Our sample is in no obvious sense representative, but it does contain randomly sampled graphs that imply OMEs with a large variety of different maximum clique sizes.

We used the OME generated from the independence constraints that a sampled 12-DAG implied as the knowledge graph at the outset of our sequence of experiments. We recorded the size of the largest clique in the OME, and then performed a sequence of experiments, in which for each experiment the intervention set was determined by OPTINTER. In each experiment we assumed that we had access to an oracle for the independence constraints of the manipulated distribution, and we updated our knowledge graph after each experiment given the new independence constraints. We recorded the number of experiments necessary to uniquely determine the DAG.

In the plot below we show the number of experiments required to uniquely determine the causal structure given the OME of 12-DAGs with various maximum clique sizes (x-axis). We plot the mean and the maximum number of experiments that were required, and show how they compare against the conjectured number of experiments. (Theoretically the minimum number of experiments is always 1 for maximal cliques greater or equal to two in an OME.)



The plot shows that the conjecture is not violated and that it in fact appears to be tight in the sense that the maximum number of experiments necessary to uniquely determine the causal graph does in fact coincide with the conjectured bound. Furthermore, we see that the mean number of experiments grows more slowly, despite the fact that we sampled very dense graphs (with 53 out of a possible 55 edges) to test maximal clique sizes of 5 and greater. OPTINTER selects variables for the intervention set randomly if there is no reason to prefer one over the other, and so the lower mean is due to "lucky" samples of intervention variables. Of course, the particular growth rate of the mean should not be taken as informative because of our sampling method. The mean is based on different sample sizes in each case. The following table shows how many OMEs each data point in the plot is based on:

MaxClqSize	1	2	3	4	5	6
# of OMEs	N/A	495	349	253	278	286
MaxClqSize	7	8	9	10	11	12
# of OMEs	242	152	123	46	1*	1*

5 DISCUSSION

So far we only considered experiments involving *multiple simultaneous* interventions on an OME. The difficulty in specifying a bound on the number of experiments given an OME, when only single (or only very few) interventions are permitted per experiment, is due to the interdependence of the orientation of two adjacent edges. For example, if we know from passive observation that the OME of the true causal graph is a chain of undirected edges,

$$X_1 \text{ —— } X_2 \text{ ——·····—— } X_N$$

then we know that this chain cannot contain any v-structures, since they would have been discovered in the passive observation. But it could contain a common cause at any vertex (except the ends). Consequently, if only a single intervention can be performed per experiment, the most efficient strategy would be to intervene on the middle vertex, resulting in a sequence of $\log_2(N)$ experiments in the worst case to recover the causal structure. OPTINTER, without constraints on $maxInter$ would perform a single experiment with every other variable in the intervention set. With $maxInter = 1$, OPTINTER might well take $N/2$ experiments. Similar arguments apply to tree structures and particular planar networks made up of triangles or diamond shapes. Ultimately, if there are many dependencies between the directions of edges, then these can be exploited to improve the efficiency of discovery, but greedy procedures like OPTINTER are suboptimal. In general the number of experiments necessary and sufficient to discover the causal graph given an OME when only single interventions can be performed per experiment, is bounded by the $\sum_i(|C_i|-1)$, where the C_i are *non-overlapping* maximal cliques. But we

have no results on how tight this bound is, nor do we have a method, other than brute force, to compute the appropriate intervention sets. For similar reasons, the intervention sets computed by OPTINTER are not minimal with regard to the number of variables subject to an intervention in one experiment or over the entire sequence of experiments.

Computing the appropriate intervention set given a knowledge graph is closely related to the MAX-CUT problem, which is in general NP-complete. There are approximation algorithms, with the best offering a 0.878-approximation (Goemans & Williamson 1995). The approximation MAX-CUT algorithm is, of course, not designed with the specific aim to orient edges in cliques. Hence, an approximate MAX-CUT might not be sufficient to guarantee the conjectured bound (even if true). The OPTINTER algorithm is a greedy algorithm that selects the intervention set specifically in light of the above conjectured bound. But since OPTINTER is computationally expensive (due to the clique search), a MAX-CUT approximation algorithm may be a better choice for large graphs even if the guarantees supplied may be weaker than those of OPTINTER.

6 CONCLUSION

We have provided an algorithm for the computation of intervention sets based on the qualitative structural constraints on hypothesis spaces that can be represented by knowledge graphs. Knowledge graphs provide a concise representation of hypotheses that characterize the underdetermination of causal structures from passive observational data (observational Markov equivalence classes) and underdetermination resulting from sequences of experiments. We conjecture that the number of experiments sufficient and in the worst case necessary to discover the true causal structure in an OME is a function of the largest undirected clique in the OME, and we have shown why this conjecture cannot be generalized to all knowledge graphs. In simulations we have shown that our OPTINTER algorithm determines intervention sets that generate sequences of experiments that satisfy the conjecture. As a result, we have a search procedure that is adaptive and therefore much more efficient (in the number of experiments) than procedures that satisfy worst case bounds for the space of all DAGs. Consequently, we think this algorithm is much more relevant to actual scientific practice.

Acknowledgements

I am very grateful to Oleg Pikhurko for pointing me to Folkman's Theorem. This research was funded by a fellowship from the James S. McDonnell Foundation.

References

D. M. Chickering (2002). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507-554.

G. Cooper and C. Yoo (1999). Causal discovery from a mixture of data. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence*, 116-125, San Francisco, CA.: Morgan Kaufmann.

F. Eberhardt, C. Glymour, and R. Scheines (2005). On the number of experiments sufficient and in the worst case necessary to identify all causal relations among N variables. In F. Bacchus and T. Jaakkola (ed.), *Proceedings of the 21 st Conference on Uncertainty and Artificial Intelligence*, 178-184. AUAI Press, Corvallis, Oregon.

F. Eberhardt (2007). *Causation and Intervention*, PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

J. Folkman (1970). Graphs with monochromatic complete subgraphs in every edge coloring. *SIAM Journal on Applied Mathematics*, 18, No. 1:19-24.

M. X. Goemans and D. P. Williamson (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42,6:1115-1145.

C. Meek (1995). Causal inference and causal explanation with background knowledge. In *Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence*, 403-418, Montreal, QU.: Morgan Kaufmann.

S. Meganck, B. Manderick, and P. Leray (2005). A decision theoretic approach to learning bayesian networks. Technical report, Vrije Universiteit Brussels.

K. P. Murphy (2001). Active learning of causal bayes net structure. Technical report, Department of Computer Science, U.C. Berkeley.

J. Pearl (2000). *Causality.* Oxford University Press.

S. Shimizu, P. O. Hoyer, A. Hyvärinen, and A. J. Kerminen (2006). A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research* 7:2003-2030.

P. Spirtes, C. Glymour, and R. Scheines (2000). *Causation, Prediction and Search.* MIT Press, 2 edition.

S. Tong and D. Koller (2001). Active learning for structure in bayesian networks. In *Proceedings of the International Joint Conference on Artificial Intelligence.*

Gibbs Sampling in Factorized Continuous-Time Markov Processes

Tal El-Hay **Nir Friedman**
School of Computer Science
The Hebrew University
{tale,nir}@cs.huji.ac.il

Raz Kupferman
Institute of Mathematics
The Hebrew University
raz@math.huji.ac.il

Abstract

A central task in many applications is reasoning about processes that change over continuous time. *Continuous-Time Bayesian Networks* is a general compact representation language for multi-component continuous-time processes. However, exact inference in such processes is exponential in the number of components, and thus infeasible for most models of interest. Here we develop a novel Gibbs sampling procedure for multi-component processes. This procedure iteratively samples a trajectory for one of the components given the remaining ones. We show how to perform *exact* sampling that adapts to the natural time scale of the sampled process. Moreover, we show that this sampling procedure naturally exploits the structure of the network to reduce the computational cost of each step. This procedure is the first that can provide asymptotically unbiased approximation in such processes.

1 Introduction

In many applications, we reason about processes that evolve over time. Such processes can involve short time scales (e.g., the dynamics of molecules) or very long ones (e.g., evolution). In both examples, there is no obvious discrete "time unit" by which the process evolves. Rather, it is more natural to view the process as changing in a continuous time: the system is in some state for a certain duration, and then transitions to another state. The language of *continuous-time Markov processes* (CTMPs) provides an elegant mathematical framework to reason about the probability of trajectories of such systems (Gardiner, 2004). We consider Markov processes that are homogeneous in time and have a finite state space. Such systems are fully determined by the state space S, the distribution of the process at the initial time, and a description of the dynamics of the process. These dynamics are specified by a *rate matrix* $\mathbb{Q}$, whose off-diagonal entries $q_{\boldsymbol{a},\boldsymbol{b}}$ are exponential rate intensities for transitioning from state $\boldsymbol{a}$ to $\boldsymbol{b}$. Intuitively, we can think of the entry $q_{\boldsymbol{a},\boldsymbol{b}}$ as the rate parameter of an exponential distribution whose value is the duration of time spent in state $\boldsymbol{a}$ before transitioning to $\boldsymbol{b}$.

In many applications, the state space is of the form of a product space $S = S_1 \times S_1 \times \cdots \times S_M$, where M is the number of *components* (such processes are called multi-component). Even if each of the S_i is of low dimension, the dimension of the state space is exponential in the number of components, which poses representational and computational difficulties. Recently, Nodelman et al. (2002) introduced the representation language of *continuous-time Bayesian networks* (CTBNs), which provides a factorized, component-based representation of CTMPs: each component is characterized by a conditional CTMP dynamics, which describes its local evolution as a function of the current state of its parents in the network. This representation is natural for describing systems with a sparse structure of local influences between components.

For most applications of such CTMP models, we need to perform inference to evaluate the posterior probability of various queries given evidence. Exact inference requires exponentiation of the rate matrix $\mathbb{Q}$. As the rate matrix is exponential in the number of components, exact computations are infeasible for more than a few components. Thus, applications of factored CTMPs require the use of approximate inference.

In two recent works Nodelman et al. (2005) and Saria et al. (2007) describe approximate inference procedures based on Expectation Propagation, a variational approximation method (Minka, 2001; Heskes and Zoeter, 2002). These approximation procedures perform local propagation of messages between components (or sub-trajectories of components) until convergence. Such procedures can be quite efficient, however they can also introduce a systematic error in the approximation (Fan and Shelton, 2008).

More recently, Fan and Shelton (2008) introduced a procedure that employs importance sampling and particle filtering to sample trajectories from the network. Such a stochastic sampling procedure has anytime properties as collecting more samples leads to more accurate approximation. However, since this is an importance sampler, it has limited capabilities to propagate evidence "back" to influence the sampling of earlier time steps. As a result, when the evidence is mostly at the end of the relevant time inter-

val, and is of low probability, the procedure requires many samples. A related importance sampler was proposed by Ng et al. (2005) for monitoring a continuous time process.

In this paper we introduce a new stochastic sampling procedure for factored CTMPs. The goal is to sample random system trajectories from the posterior distribution. Once we have multiple independent samples from this distribution we can approximate the answer to queries about the posterior using the empirical distribution of the samples. The challenge is to sample from the posterior. While generative sampling of a CTMP is straightforward, sampling given evidence is far from trivial, as evidence modifies the posterior probability of earlier time points.

Markov Chain Monte Carlo (MCMC) procedures circumvent this problem by sampling a stochastic sequence of system states (trajectories in our models) that will eventually be governed by the desired posterior distribution. Here we develop a Gibbs sampling procedure for factored CTMPs. This procedure is initialized by setting an arbitrary trajectory which is consistent with the evidence. It then alternates between randomly picking a component X_i and sampling a trajectory from the distribution of X_i conditioned on the trajectories of the other components and the evidence. This procedure is reminiscent of *block Gibbs sampling* (Gilks et al., 1996) as we sample an entire trajectory rather than a single random variable in each iteration. However, in our approach we need to sample a continuous trajectory.

The crux of our approach is in the way we sample a trajectory for a single component from a process that is conditioned on trajectories of the other components. While such a process is Markovian, it is not homogeneous as its dynamics depends on trajectories of its Markov Blanket as well as on past and present evidence. We show that we can perform exact sampling by utilizing this Markovian property, and that the cost of this procedure is determined by the complexity of the current trajectories and the sampled one, and not by a pre-defined resolution parameter. This implies that the computational time adapts to the complexity of the sampled object.

2 Continuous-Time Bayesian Networks

In this section we briefly review the CTBN model (Nodelman et al., 2002). Consider an M-component Markov process

$$\boldsymbol{X}^{(t)} = (X_1^{(t)}, X_2^{(t)}, \dots X_M^{(t)})$$

with state space $S = S_1 \times S_2 \times \cdots \times S_M$.

A notational convention: vectors are denoted by boldface symbols, e.g., $\boldsymbol{X}, \boldsymbol{a}$, and matrices are denoted by blackboard style characters, e.g., $\mathbb{Q}$. The states in S are denoted by vectors of indexes, $\boldsymbol{a} = (a_1, \dots, a_M)$. The indexes $1 \le i, j \le M$ are used to enumerate the components. We use the notation $\boldsymbol{X}^{(t)}$ and $X_i^{(t)}$ to denote a random variable at time t. We will use $\boldsymbol{X}^{[s,t]}$, $\boldsymbol{X}^{(s,t]}$, $\boldsymbol{X}^{[s,t)}$, to denote the state of $\boldsymbol{X}$ in the closed and semi-open intervals from s to t.

The dynamics of a time-homogeneous continuous-time Markov process are fully determined by the *Markov transition function*,

$$p_{\boldsymbol{a},\boldsymbol{b}}(t) = \Pr(\boldsymbol{X}^{(t+s)} = \boldsymbol{b} | \boldsymbol{X}^{(s)} = \boldsymbol{a}),$$

where time-homogeneity implies that the right-hand side does not depend on s. Provided that the transition function satisfies certain analytical properties (continuity, and regularity; see Chung (1960)) the dynamics are fully captured by a constant matrix $\mathbb{Q}$—the *rate*, or *intensity matrix*—whose entries $q_{\boldsymbol{a},\boldsymbol{b}}$ are defined by

$$q_{\boldsymbol{a},\boldsymbol{b}} = \lim_{h \downarrow 0} \frac{p_{\boldsymbol{a},\boldsymbol{b}}(h) - \delta_{\boldsymbol{a},\boldsymbol{b}}}{h},$$

where $\delta_{\boldsymbol{a},\boldsymbol{b}}$ is a multivariate Kronecker delta.

A Markov process can also be viewed as a generative process: The process starts in some state $\boldsymbol{a}$. After spending a finite amount of time at $\boldsymbol{a}$, it transitions, at a random time, to a random state $\boldsymbol{b} \neq \boldsymbol{a}$. The transition times to the various states are exponentially distributed, with rate parameters $q_{\boldsymbol{a},\boldsymbol{b}}$. The diagonal elements of $\mathbb{Q}$ are set such that each row sums up to zero.

The time-dependent probability distribution, $\boldsymbol{p}(t)$, whose entries are defined by

$$p_{\boldsymbol{a}}(t) = \Pr(\boldsymbol{X}^{(t)} = \boldsymbol{a}), \qquad \boldsymbol{a} \in S,$$

satisfies the so-called *forward*, or *master*, *equation*,

$$\frac{d\boldsymbol{p}}{dt} = \mathbb{Q}^T \boldsymbol{p}. \tag{1}$$

Thus, using the $\mathbb{Q}$ matrix, we can write the Markov transition function as

$$p_{\boldsymbol{a},\boldsymbol{b}}(t) = \left[\exp(t\mathbb{Q})\right]_{\boldsymbol{a},\boldsymbol{b}},$$

that is, as the $\boldsymbol{a}, \boldsymbol{b}$ entry in the matrix resulting from exponentiating $\mathbb{Q}$ (using matrix exponentiation).

It is important to note that the master Eq. (1) encompasses all the statistical properties of the Markov process. There is a one-to-one correspondence between the description of a Markov process by means of a master equation, and by means of a "pathwise" characterization (up to stochastic equivalence of the latter; see Gikhman and Skorokhod (1975)).

Continuous-time Bayesian Networks provide a compact representation of multi-component Markov processes by incorporating two assumptions: (1) every transition involves a single component; (2) each component undergoes transitions at a rate which depends only on the state of a subsystem of components.

Formally, the structure of a CTBN is defined by assigning to each component i a set of indices $\mathrm{Par}(i) \subseteq$

$\{1,\ldots,M\} \setminus \{i\}$. With each component i, we associate a *conditional rate matrix* $\mathbb{Q}^{i|\,\mathrm{Par}(i)}$ with entries $q^{i|\,\mathrm{Par}(i)}_{a_i,b_i|\boldsymbol{u}_i}$ where a_i and b_i are states of X_i and $\boldsymbol{u}_i$ is a state of $\mathrm{Par}(i)$. This matrix defines the rate of X_i as a function of the state of its parents. Thus, when the parents of X_i change state, the rates governing its transition can change.

The formal semantics of CTBNs is in terms of a joint rate matrix for the whole process. This rate matrix is defined by combining the conditional rate matrices

$$q_{\boldsymbol{a},\boldsymbol{b}} = \sum_{i=1}^{M} \left(q^{i|\,\mathrm{Par}(i)}_{a_i,b_i|\,\mathrm{P}_i(\boldsymbol{a})} \prod_{j \neq i} \delta_{a_j,b_j} \right). \qquad (2)$$

where $\mathrm{P}_i(\boldsymbol{a})$ is a projection operator that project a complete assignment $\boldsymbol{a}$ to an assignment to the $\mathrm{Par}(i)$ components. Eq. (2) is, using the terminology of Nodelman et al. (2002), the "amalgamation" of the M conditional rate matrices. Note the compact representation, which is valid for both diagonal and off-diagonal entries. It is also noteworthy that amalgamation is a summation, rather than a product; indeed, independent exponential rates are additive. If, for example, every component has d possible values and k parents, the rate matrix requires only $Md^{k+1}(d-1)$ parameters, rather than $d^M(d^M-1)$.

The dependency relations between components can be represented graphically as a directed graph, $\mathcal{G}$, in which each node corresponds to a component, and each directed edge defines a parent-child relation. A CTBN consists of such a graph, supplemented with a set of M conditional rate matrices $\mathbb{Q}^{i|\,\mathrm{Par}(i)}$. The graph structure has two main roles: (i) it provides a data structure to which parameters are associated; (ii) it provides a qualitative description of dependencies among the various components of the system. The graph structure also reveals conditional independencies between sets of components (Nodelman et al., 2002).

Notational conventions: Full trajectories and observed pointwise values of components are denoted by lower case letters indexed by the relevant time intervals, e.g., $x_i^{(t)}$, $x_i^{[s,t]}$. We will use $\Pr(x_i^{(t)})$ and $\Pr(x_i^{[s,t]})$ as shorthands for $\Pr(X_i^{(t)} = x_i^{(t)})$ and $\Pr(X_i^{[s,t]} = x_i^{[s,t]})$.

It should be emphasized that even though CTBNs provide a succinct representation of multi-component processes, any inference query still requires the exponentiation of the full $d^M \times d^M$ dimensional rate matrix $\mathbb{Q}$. For example, given the state of the system at times 0 and T, the *Markov bridge* formula is

$$\Pr(\boldsymbol{X}^{(t)} = \boldsymbol{a} | \boldsymbol{x}^{(0)}, \boldsymbol{x}^{(T)}) = \frac{[\exp(t\mathbb{Q})]_{\boldsymbol{x}^{(0)},\boldsymbol{a}} [\exp((T-t)\mathbb{Q})]_{\boldsymbol{a},\boldsymbol{x}^{(T)}}}{[\exp(T\mathbb{Q})]_{\boldsymbol{x}^{(0)},\boldsymbol{x}^{(T)}}}.$$

It is the premise of this work that such expressions cannot be computed directly, thus requiring approximation algorithms.

3 Sampling in a Two Component Process

3.1 Introduction

We will start by addressing the task of sampling from a two components process. The generalization to multi-component processes will follow in the next section.

Consider a two-component CTBN, $\boldsymbol{X} = (X, Y)$, whose dynamics is defined by conditional rates $\mathbb{Q}^{X|Y}$ and $\mathbb{Q}^{Y|X}$ (that is, X is a parent of Y and Y is a parent of X). Suppose that we are given partial evidence about the state of the system. This evidence might contain point observations, as well as continuous observations in some intervals, of the states of one or two components. Our goal is to sample a trajectory of (X, Y) from the joint posterior distribution.

The approach we take here is to use a Gibbs sampler (Gilks et al., 1996) over trajectories. In such a sampler, we initialize X and Y with trajectories that are consistent with the evidence. Then, we randomly either sample a trajectory of X given the entire trajectory of Y and the evidence on X, or sample a trajectory of Y given the entire trajectory of X and the evidence on Y. This procedure defines a random walk in the space of (X, Y) trajectories. The basic theory of Gibbs sampling suggests that this random walk will converge to the distribution of X, Y given the evidence.

To implement such a sampler, we need to be able to sample the trajectory of one component given the entire trajectory of the other component and the evidence. Suppose, we have a fully observed trajectory on Y. In this case, observations on X at the extremities of some time interval statistically separate this interval from the rest of trajectory. Thus, we can restrict our analysis to the following situation: the process is restricted to a time interval $[0, T]$ and we are given observations $X^{(0)} = x^{(0)}$ and $X^{(T)} = x^{(T)}$, along with the entire trajectory of Y in $[0, T]$. The latter consists of a sequence of states $(y_0, \ldots, y_K)$ and transition times $(\tau_0 = 0, \tau_1, \ldots, \tau_K, \tau_{K+1} = T)$. An example of such scenario is shown in Figure 1(a). The entire problem is now reduced to the following question: how can we sample a trajectory of X in the interval $(0, T)$ from its posterior distribution?

To approach this problem we exploit the fact that *the sub-process X given that $Y^{[0,T]} = y^{[0,T]}$ is Markovian* (although non-homogeneous in time):

Proposition 3.1: *The following Markov property holds for all $t > s$,*

$$\Pr(X^{(t)} \mid x^{[0,s]}, x^{(T)}, y^{[0,T]}) = \Pr(X^{(t)} \mid x^{(s)}, x^{(T)}, y^{[s,T]}).$$

3.2 Time Granularized Process

Analysis of such process requires reasoning about a continuum of random variables. A natural way of doing so is to perform the analysis in discrete time with a finite time granularity h, and examine the behavior of the system when we take $h \downarrow 0$.

To do so, we introduce some definitions. Suppose $\Pr$ is the probability function associated with a continuous-time Markov process with rate matrix $\mathbb{Q}$. We define the *h-coarsening* of $\Pr$ to be $\Pr_h$, a distribution over the random variables $\boldsymbol{X}^{(0)}, \boldsymbol{X}^{(h)}, \boldsymbol{X}^{(2h)}, \ldots$ which is defined by the dynamics

$$\Pr_h(\boldsymbol{X}^{(t+h)} = \boldsymbol{b} \mid \boldsymbol{X}^{(t)} = \boldsymbol{a}) = \delta_{\boldsymbol{a},\boldsymbol{b}} + h \cdot q_{\boldsymbol{a},\boldsymbol{b}},$$

which is the Taylor expansion of $[\exp(t\mathbb{Q})]_{\boldsymbol{a},\boldsymbol{b}}$, truncated at the linear term. When $h < \min_{\boldsymbol{a}}(-1/q_{\boldsymbol{a},\boldsymbol{a}})$, $\Pr_h$ is a well-defined distribution.

We would like to show that the measure $\Pr_h(A)$ of an event A converges to $\Pr(A)$ when $h \downarrow 0$. To do so, however, we need to define the h-coarsening of an event. Given a time point t, define $\lfloor t \rfloor_h$ and $\lceil t \rceil_h$ to be the rounding down and up of t to the nearest multiple of h. For point events we define $[\![\boldsymbol{X}^{(t)} = \boldsymbol{a}]\!]_h$ to be the event $\boldsymbol{X}^{(\lfloor t \rfloor_h)} = \boldsymbol{a}$, and $[\![\boldsymbol{X}^{(t^+)} = \boldsymbol{a}]\!]_h$ to the event $\boldsymbol{X}^{(\lceil t \rceil_h)} = \boldsymbol{a}$. For an interval event, we define $[\![\boldsymbol{X}^{(s,t]} = \boldsymbol{a}_{(s,t]}]\!]_h$ to be the event $\boldsymbol{X}^{(\lceil s \rceil_h)} = \boldsymbol{a}_{\lceil s \rceil_h}, \boldsymbol{X}^{(\lceil s \rceil_h + h)} = \boldsymbol{a}_{\lceil s \rceil_h + h}, \ldots, \boldsymbol{X}^{(\lfloor t \rfloor_h)} = \boldsymbol{a}_{\lfloor t \rfloor_h}$. Similarly, we can define the coarsening of events over only one component and composite events.

Note that the probability of any given trajectory tends to zero as $h \to 0$. The difficulty in working directly in the continuous-time formulation is that we condition on events that have zero probability. The introduction of a granularized process allows us to manipulate well-defined conditional probabilities, which remain finite as $h \to 0$.

Theorem 3.2: Let A and B be point, interval, or a finite combination of such events. Then

$$\lim_{h \downarrow 0} \Pr_h([\![A]\!]_h \mid [\![B]\!]_h) = \Pr(A \mid B)$$

From now on, we will drop the $[\![A]\!]_h$ notation, and assume it implicitly in the scope of $\Pr_h()$.

A simple minded approach to solve our problem is to work with a given finite h and use discrete sampling to sample trajectories in the coarsened model (thus, working with a *dynamical Bayesian network*). If h is sufficiently small this might be a reasonable approximation to the desired distribution. However, this approach suffers from sub-optimality due to this fixed time granularity — a too coarse granularity leads to inaccuracies, while a too fine granularity leads to computational overhead. Moreover, when different components evolve at different rates, this trade-off is governed by the fastest component.

3.3 Sampling a Continuous-Time Trajectory

To avoid the trade-offs of fixed time granularity we exploit the fact that while a single trajectory is defined over infinite time points, it involves only a finite number of transitions in a finite interval. Therefore, instead of sampling states at different time points, we only sample a finite sequence of transitions. The Markovian property of the conditional process X enables doing so using a sequential procedure.

Our procedure starts by sampling the first transition time. It then samples the new state the transition leads to. As this new sample point statistically separates the remaining interval from the past, we are back with the initial problem yet with a shorter interval. We repeat these steps until the entire trajectory is sampled; it terminates once the next transition time is past the end of the interval.

Our task is to sample the first transition time and the next state, conditioned on $X^{(0)} = x^{(0)}$, $X^{(T)} = x^{(T)}$ as well as the entire trajectory of Y in $[0, T]$. To sample this transition time, we first define the conditional cumulative distribution function $F(t)$ that X stays in the initial state for a time less than t:

$$F(t) = 1 - \Pr\left(X^{(0,t]} = x^{(0)} | x^{(0)}, x^{(T)}, y^{[0,T]}\right) \quad (3)$$

If we can evaluate this function, then we can sample the first transition time τ by inverse transform sampling — we draw ξ from a uniform distribution in the interval $[0, 1]$, and set $\tau = F^{-1}(\xi)$; see Figure 1a,b.

The Markov property of the conditional process allows us to decompose the probability that X remains in its initial state until time t. Denoting the probability of Y's trajectory and of X remaining in its initial state until time t by

$$p^{\text{past}}(t) = \Pr(X^{(0,t]} = x^{(0)}, y^{(0,t]} | x^{(0)}, y^{(0)}),$$

and the probability of future observations given the state of (X_t, Y_t) by

$$p^{\text{future}}_x(t) = \Pr(x^{(T)}, y^{(t,T]} | X^{(t)} = x, y^{(t)}).$$

We can then write the probability that X is in state $x^{(0)}$ until t as

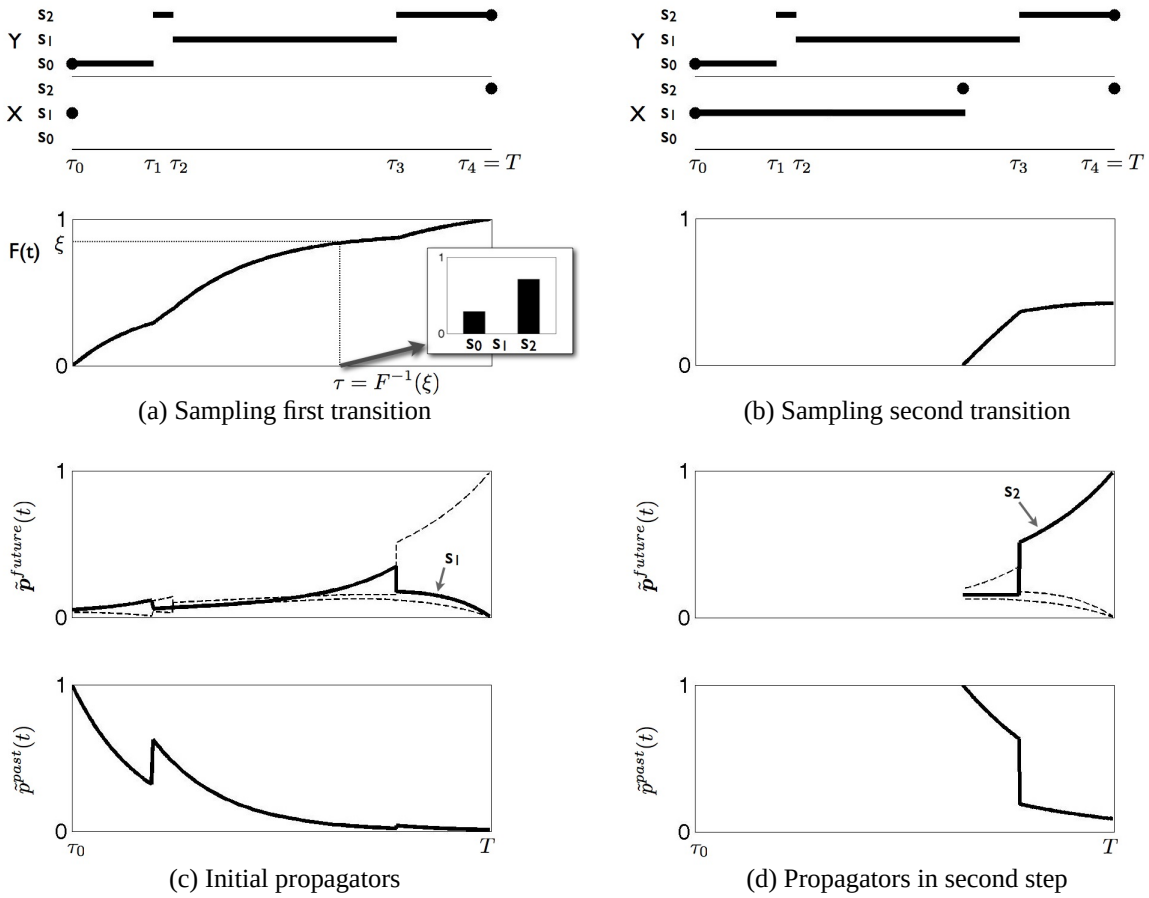
$$\Pr\left(X^{(0,t]} = x^{(0)} | x^{(0)}, x^{(T)}, y^{[0,T]}\right) = \frac{p^{\text{past}}(t) \cdot p^{\text{future}}_{x^{(0)}}(t)}{p^{\text{future}}_{x^{(0)}}(0)}. \quad (4)$$

Lamentably, while the reasoning we just described is seemingly correct, all the terms in Eq. (4) are equal to 0, since they account for the probability of Y's trajectory. However, as we shall see, if we evaluate this equation carefully we will be able to define it with terms that decompose the problem in a similar manner.

To efficiently compute these terms we exploit the fact that although the process is not homogeneous, the dynamics of the joint process within an interval $[\tau_k, \tau_{k+1})$, in which Y has a fixed value y_k, is characterized by a *single* unnormalized rate matrix whose entries depend on y_k. This allows us to adopt a *forward-backward* propagation scheme. We now develop the details of these propagations.

3.4 Computing $p^{\text{past}}(t)$

We begin with expressing $p^{\text{past}}(t)$ as a product of local terms. Recall that $p^{\text{past}}(t)$ is the probability that X is constant until time t. We denote by $p^{\text{past}}_h(t)$ the h-coarsened version of $p^{\text{past}}(t)$.



(a) Sampling first transition

(b) Sampling second transition

(c) Initial propagators

(d) Propagators in second step

Figure 1: Illustration of sampling of a single component with three states. (a) Top panel: sampling scenario, with a complete trajectory for Y, that has four transitions at $\tau_1, \ldots, \tau_4$, and point evidence on X at times 0 and T. Bottom panel: the cumulative distribution $F(t)$, that X changes states before time t given this evidence. We sample the next transition time by drawing ξ from a uniform distribution and setting $\tau = F^{-1}(\xi)$. Note that as $x^{(0)} \neq x^{(T)}$, $F(T) = 1$. The bar graph represents the conditional distribution of the next state, given a transition at time τ. (b) Same sampling procedure for the second transition. Here $F(T) < 1$ since it is not necessary for X to change its state. (c and d) The two components used in computing $1 - F(t)$: $\tilde{p}^{\text{past}}(t)$ the probability that X stays with a constant value until time t and Y has the observed trajectory until this time; and $\tilde{p}_t^{\text{future}}(x)$ the probability that X transition's from state x at t to its observed state at time T and Y follows its trajectory from t to T.

To characterize the dynamics within intervals (τ_k, τ_{k+1}) we define *constant propagator functions*

$$\phi_{h,x}^{y}(\Delta t) = \Pr_h(X^{(t,t+\Delta t]} = x, Y^{(t,t+\Delta t]} = y | X^{(t)} = x, Y^{(t)} = y)$$

These functions determine the probability that $X = x$ and $Y = y$ throughout an interval of length Δt if they start with these values.

At time τ_{k+1} the Y component changes it value from y_k to y_{k+1}. The transition probability at this point is $h \cdot q^{Y|X}_{y_k, y_{k+1}|x^{(0)}}$. Thus, from the Markov property of the joint process it follows that for $t \in (\tau_k, \tau_{k+1})$

$$p_h^{\text{past}}(t) = \left[\prod_{l=0}^{k-1} \phi_{h,x^{(0)}}^{y_l}(\Delta_l) \cdot q^{Y|X}_{y_l, y_{l+1}|x^{(0)}} \cdot h \right] \phi_{h,x^{(0)}}^{y_k}(t - \tau_k)$$

where $\Delta_l = \tau_{l+1} - \tau_l$.

To compute the constant propagator functions, we realize that in each step within the interval $(s, t]$ the state does not change. Thus,

$$\phi_{h,x}^{y}(\Delta t) = [1 + h \cdot (q^{X|Y}_{x,x|y} + q^{Y|X}_{y,y|x})]^{\frac{\lfloor \Delta t \rfloor_h}{h}}$$

We define

$$\phi_{x}^{y}(\Delta t) = \lim_{h \downarrow 0} \phi_{h,x}^{y}(\Delta t) = e^{(\Delta t)(q^{X|Y}_{x,x|y} + q^{Y|X}_{y,y|x})}$$

We conclude that if

$$\tilde{p}^{\text{past}}(t) = \left[\prod_{l=0}^{k-1} \phi^{y_l}_{x^{(0)}}(\Delta_l) \cdot q^{Y|X}_{y_l,y_{l+1}|x^{(0)}}\right] \phi^{y_k}_{x^{(0)}}(t-\tau_k),$$

then for $t \in (\tau_k, \tau_{k+1})$

$$\lim_{h\downarrow 0} \frac{p^{\text{past}}_h(t)}{h^k} = \tilde{p}^{\text{past}}(t)$$

3.5 Computing $p^{\text{future}}_x(t)$

We now turn to computing $p^{\text{future}}_x(t)$. Unlike the previous case, here we need to compute this term for every possible value of x. We do so by backward dynamic programing (reminiscent of backward messages in HMMs).

We denote by $\boldsymbol{p}^{\text{future}}_h(t)$ a vector with entries $p^{\text{future}}_{h,x}(t)$. Note that, $\boldsymbol{p}^{\text{future}}_h(T) = \boldsymbol{e}_{x^{(T)}}$ where $\boldsymbol{e}_x$ is the unit vector with 1 in position x. Next, we define a *propagator matrix* $\mathbb{S}^y_h(\Delta t)$ with entries

$$s^y_{h,a,b}(\Delta t) = \Pr_h(X^{(t+\Delta t)} = b, Y^{(t,t+\Delta t]} = y | X^{(t)} = a, Y^{(t)} = y)$$

This matrix provides the dynamics of X in an interval where Y is constant. We can use it to compute the probability of transitions between states of X in the intervals $(\tau_k, \tau_{k+1}]$, for every $\tau_k < s < t < \tau_{k+1}$

$$\boldsymbol{p}^{\text{future}}_h(s) = \mathbb{S}^{y_k}_h(t-s)\boldsymbol{p}^{\text{future}}_h(t)$$

At transition points τ_k we need to take into account the probability of a change. To account for such transitions, we define a diagonal matrix $\mathbb{T}^{y,y'}$ whose (a,a) entry is $q^{Y|X}_{y,y'|a}$. Using this notation and the Markov property of the joint process the conditional probability of future observations for $\tau_k \le t \le \tau_{k+1}$ is

$$\boldsymbol{p}^{\text{future}}_h(t) = \mathbb{S}^{y_{k-1}}_h(\tau_{k+1}-t)\left[\prod_{l=k+1}^{K} h\mathbb{T}^{y_l,y_{l+1}}\mathbb{S}^y_h(\Delta_l)\right]\boldsymbol{e}_{x^{(T)}}$$

It remains to determine the form of the propagator matrix. At time granularity h, we can write the probability of transitions between states of X while $Y = y$ as a product of transition matrices. Thus,

$$\mathbb{S}^y_h(\Delta t) = (I + h\cdot \mathbb{R}_{X|y})^{\frac{\lfloor \Delta t \rfloor_h}{h}}$$

where $\mathbb{R}^{X|y}$ is the matrix with entries

$$r^{X|y}_{a,b} = \begin{cases} q^{X|Y}_{a,b|y} & a \neq b \\ q^{X|Y}_{a,a|y} + q^{Y|X}_{y,y|a} & a = b \end{cases}$$

We now can define

$$\mathbb{S}^y(\Delta t) = \lim_{h\downarrow 0} \mathbb{S}^y_h(\Delta t) = e^{(\Delta t)\mathbb{R}^{X|y}}$$

This terms is similar to transition matrix of a Markov process. Note, however that $\mathbb{R}$ is not a stochastic rate matrix, as the rows do not sum up to 0. In fact, the sum of the rows in negative, which implies that the entries in $\mathbb{S}^y_h(\Delta t)$ tend to get smaller with Δt. This matches the intuition that this term should capture the probability of the evidence that $Y = y$ for the whole interval.

To summarize, if we define for $t \in (\tau_k, \tau_{k+1})$

$$\tilde{\boldsymbol{p}}^{\text{future}}(t) = \mathbb{S}^{y_{k-1}}(\tau_{k+1}-t)\left[\prod_{l=k+1}^{K} \mathbb{T}^{y_l,y_{l+1}}\mathbb{S}^y(\Delta_l)\right]\boldsymbol{e}_{x^{(T)}},$$

then

$$\lim_{h\downarrow 0} \frac{\boldsymbol{p}^{\text{future}}_h(t)}{h^{K-k}} = \tilde{\boldsymbol{p}}^{\text{future}}(t)$$

3.6 Putting it All Together

Based on the above arguments.

$$\Pr_h\left(X^{(0,t]} = x^{(0)} | x^{(0)}, x^{(T)}, y^{[0,T]}\right) = \frac{p^{\text{past}}_h(t) p^{\text{future}}_{h,x^{(0)}}(t)}{p^{\text{future}}_{h,x^{(0)}}(0)}$$

Now, if $t \in (\tau_k, \tau_{k+1})$, then

$$\begin{aligned} &\Pr\left(X^{(0,t]} = x^{(0)} | x^{(0)}, x^{(T)}, y^{[0,T]}\right) \\ &= \lim_{h\downarrow 0} \frac{p^{\text{past}}_h(t) p^{\text{future}}_{h,x^{(0)}}(t)}{p^{\text{future}}_{h,x^{(0)}}(0)} \\ &= \lim_{h\downarrow 0} \frac{[h^{-k} p^{\text{past}}_h(t)][h^{-(K-k)} p^{\text{future}}_{h,x^{(0)}}(t)]}{h^{-K} p^{\text{future}}_{h,x^{(0)}}(0)} \\ &= \frac{\tilde{p}^{\text{past}}(t)\tilde{p}^{\text{future}}_{x^{(0)}}(t)}{\tilde{p}^{\text{future}}_{x^{(0)}}(0)} \end{aligned}$$

Thus, in both numerator and denominator we must account for the observation of K transitions of Y, which have probability of $o(h^K)$. Since these term cancels out, we remain with the conditional probability over the event of interest.

3.7 Forward Sampling

To sample an entire trajectory we first compute $\tilde{\boldsymbol{p}}^{\text{future}}(t)$ only at transition times from the final transition to the start.

We sample the first transition time by drawing a random value ξ from a uniform distribution in $[0,1]$. Now we find τ such that $F(\tau) = \xi$ in two steps: First, we sequentially search for the interval $[\tau_k, \tau_{k+1}]$ such that $F(\tau_k) \le F(\tau) \le F(\tau_{k+1})$ by propagating $\tilde{p}^{\text{past}}(t)$ forward through transition points. Second, we search the exact time point within $[\tau_k, \tau_{k+1}]$ using binary search with L steps to obtain accuracy of $2^{-L}\Delta_k$. This step requires computation of $\mathbb{S}^{y_k}(2^{-L}\Delta_k)$ and its exponents $\mathbb{S}^{y_k}(2^{-l}\Delta_k)$, $l = 1, \ldots, L-1$.

Once we sample the transition time t, we need to compute the probability of the new state of X. Using similar

arguments as the ones we discussed above, we find that

$$\Pr\left(X^{(t^+)} = x | X^{[0,t)} = x^{(0)}, X^{(t^+)} \neq x^{(0)}, y^{[0,T]}\right) = \frac{q^{X|Y}_{x^{(0)},x} \cdot \tilde{p}^{\text{future}}_x(t)}{\sum_{x' \neq x^{(0)}} q^{X|Y}_{x^{(0)},x'} \cdot \tilde{p}^{\text{future}}_{x'}(t)}.$$

Thus, we can sample the next state by using the precomputed value of $\tilde{p}^{\text{future}}_x(t)$ at t.

Once we sample a transition (time and state), we can sample the next transition in the interval $[\tau, T]$. The procedure proceeds while exploiting propagators which have already been computed. It stops when $F(T) < \xi$, i.e., the next sampled transition time is greater than T. Figure 1 illustrates the conditional distributions of the first two transitions.

4 Sampling in a Multi-Component Process

The generalization from a two-component process to a general one is relatively straightforward. At each step, we need to sample a single component X_i conditioned on trajectories in $\boldsymbol{Y} = (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_M)$. To save computations we exploit the fact that given complete trajectories over the Markov blanket of X_i, which is the component set of X_i's parents, children and its children's parents, the dynamics in X_i is independent of the dynamics of all other components (Nodelman et al., 2002).

Indeed, the structured representation of a CTBN allows computations using only terms involving the Markov blanket. To see that, we first notice that within an interval whose state is $\boldsymbol{Y}_t = \boldsymbol{y}$ the propagator matrix involves terms which depend only on the parents of X_i $q^{X_i|Y}_{a,b|\boldsymbol{y}} = q^{X_i|\text{Par}(i)}_{a,b|\boldsymbol{u}_i}$ and terms which depend on the other members of the Markov blanket,

$$q^{\boldsymbol{Y}|X_i}_{\boldsymbol{y},\boldsymbol{y}|x_i} = \sum_{j \in \text{Child}(i)} q^{X_j|\text{Par}(j)}_{x_j,x_j|\boldsymbol{u}_j} + c_{\boldsymbol{y}}$$

where $c_{\boldsymbol{y}}$ does not depend on the state of X_i. Therefore, we define the reduced rate matrix $\mathbb{R}_{X_i|\boldsymbol{v}}$:

$$r^{X_i|\text{MB}(i)}_{a,b|\boldsymbol{v}} = \begin{cases} q^{X_i|\text{Par}(i)}_{a,b|\boldsymbol{u}_i} & a \neq b \\ q^{X_i|\text{Par}(i)}_{a,a|\boldsymbol{u}_i} + \sum_{j \in \text{Child}(i)} q^{X_j|\text{Par}(j)}_{x_j,x_j|\boldsymbol{u}_j} & a = b \end{cases}$$

where, $\boldsymbol{v}$ is the projection of $\boldsymbol{y}$ to the Markov blanket. Consequently the local propagator matrix becomes

$$\mathbb{S}^{\boldsymbol{v}}(t) = \exp(t \cdot \mathbb{R}_{X_i|\boldsymbol{v}}) \quad (5)$$

Importantly, this matrix differs from $\mathbb{S}^{\boldsymbol{y}}(t)$ by a scalar factor of $\exp(t \cdot c_{\boldsymbol{y}})$. The same factor arise when replacing the term in the exponent of the constant propagator. Therefore, these terms cancel out upon normalization.

This development also shows that when sampling X_i we only care about transition points of one of the trajectories in $\text{MB}(i)$. Thus, the intervals computed in the initial backward propagation are defined by these transitions. Therefore, the complexity of the backward procedure scales with the rate of X_i and its Markov blanket.

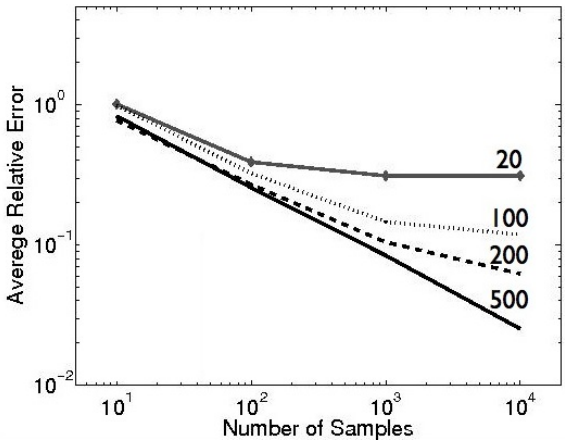


Figure 2: Relative error versus burn-in and number of samples.

5 Experimental Evaluation

We evaluate convergence properties of our procedure on a chain network presented in Fan and Shelton (2008), as well as on related networks of various sizes and parametrizations. The basic network contains 5 components, $X_0, \rightarrow X_1 \rightarrow \ldots X_4$, with 5 states each. The transition rates of X_0 suggest a tendency to cycle in 2 possible loops: $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_0$ and $s_0 \rightarrow s_3 \rightarrow s_4 \rightarrow s_0$; whereas for $i > 0$, X_i attempts to follow the state of X_{i-1} — the transition $q^{X_i|X_{i-1}}_{a,b|c}$ has higher intensity when $c = b$. The intensities of X_0 in the original network are symmetric relative to the two loops. We slightly perturbed parameters to break symmetry since the symmetry between the two loops tends to yield untypically fast convergence.

To obtain a reliable convergence assessment, we should generate samples from multiple independent chains which are initialized from an over-dispersed distribution. Aiming to construct such samples, our initialization procedure draws for each component a rate matrix by choosing an assignment to its parents from a uniform distribution and taking the corresponding conditional rate matrix. Using these matrices it samples a trajectory that is consistent with evidence independently for every component using the backward propagation-forward sampling strategy we described above.

A crucial issue in MCMC sampling is the time it takes the chain to *mix* — that is, sample from a distribution that is close to the target distribution rather than the initial distribution. It is not easy to show empirically that a chain has mixed. We examine this issue from a pragmatic perspective by asking what is the quality of the estimates based on sam-

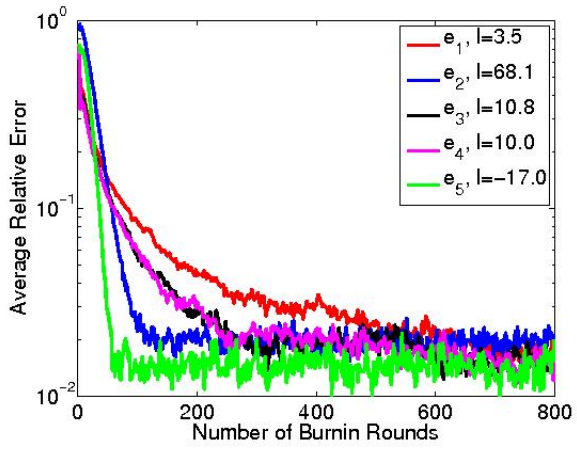


Figure 3: Error versus burn-in for different evidence sets. For each set we specify the average log-likelihood of the samples after convergence.

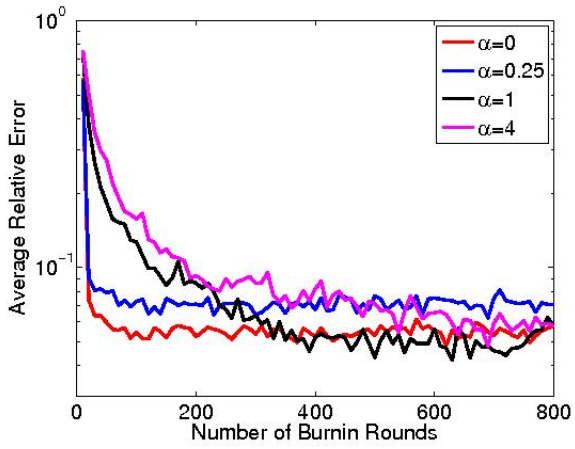


Figure 4: Effect of conditional transition probability sharpness on mixing time.

ples taken at different number of "burn-in" iterations after the initialization, where a single iteration involves sampling each of the components once. We examine the estimates of expected sufficient statistics that are required for learning CTBN's — residence time of components in states and the number of transitions given the state of the component's parent (Nodelman et al., 2003). We measure estimation quality by the *average relative error* $\sum_j \frac{|\hat{\theta}_j - \theta_j|}{\theta_j}$ where θ_j is exact value of the j'th sufficient statistics calculated using numerical integration and $\hat{\theta}_j$ is the approximation.

To make the task harder, we chose an extreme case by setting evidence $\boldsymbol{X}^{(0)} = \vec{s}_0$ (the vector of s_0), and $\boldsymbol{X}^{(3)} = (s_0, s_1, s_3, s_0, s_1)$. We then sampled the process using multiple random starting points, computed estimated expected statistics, and compared them the exact expected statistics. Figure 2 shows the behavior of the average relative error taken over all $\theta > 0.05$ versus the sample size for different number of burn-in iterations. Note that when using longer burn-in, the error decreases at a rate of $O(\sqrt{n})$, where n is the number of samples, which is what we would expect from theory, if the samples where totally independent. This implies that at this long burn-in the error due to the sampling process is smaller than the error contributed by the number of samples.

To study further the effect of evidence's likelihood, we measured error versus burn-in using 10,000 samples in our original evidence set, and four additional ones. The first additional evidence, denoted by $\boldsymbol{e}_2$ is generated by setting $\boldsymbol{X}^{(0)} = \vec{s}_0$, forward sampling a random trajectory and taking the complete trajectory of X_4 as evidence. Additional sets are: $\boldsymbol{e}_3 = \{\boldsymbol{X}^{(0)} = \vec{s}_0, \boldsymbol{X}^{(3)} = \vec{s}_0\}$; $\boldsymbol{e}_4 = \{\boldsymbol{X}^{(0)} = \vec{s}_0\}$ and an extremely unlikely case $\boldsymbol{e}_5 = \{\boldsymbol{X}^{(0)} = \vec{s}_0, X_0^{(0,3)} = s_0, \boldsymbol{X}^{(3)} = (s_0, s_1, s_3, s_0, s_1)\}$.

Figure 3 illustrates that burn-in period may vary by an order of magnitude, however it is not correlated with the log-likelihood. Note that in this specific experiment slower convergence occurs when continuous evidence is absent. The reason for this may be the existence of multiple possible paths that cycle through state zero. That is, the posterior distribution is , in a sense, multi-modal.

To further explore the effect of the posterior's landscape, we tested networks with similar total rate of transitions, but with varying level of coupling between components. Stronger coupling of components leads to a sharper joint distribution. To achieve variations in the coupling we consider variants of the chain CTBN where we set $\hat{\pi}_{a,b|y} = \frac{(q_{a,b|y})^\alpha}{\sum_{c \neq a}(q_{a,c|y})^\alpha}$ and $\hat{q}_{a,b|y} = q_{a,a|y} \cdot \hat{\pi}_{a,b|y}$ where α is a non-negative sharpness parameter As $\alpha \to 0$ the network becomes smoother, which reduces coupling between components. However, the stationary distribution is not tending to a uniform one because we do not alter the diagonal elements. Figure 4 shows convergence behavior for different values of α where estimated statistics are averaged over 1,000 samplers. As we might expect, convergence is faster as the network becomes smoother.

Next we evaluated the scalability of the algorithm by generating networks containing additional components with an architecture similar to the basic chain network. As exact inference is infeasible in such networks we measured relative error versus estimations taken from long runs. Specifically, for each N, we generated 1000 samples by running 100 independent chains and taking samples after 10,000 rounds as well as additional 9 samples from each chain every 1,000 rounds. Using these samples we estimated the target sufficient statistics. To avoid averaging different numbers of components, we compared the relative error in the estimate of 5 components for networks of different sizes. Figure 5 shows the results of this experiment. As we can see, convergence rates decay moderately

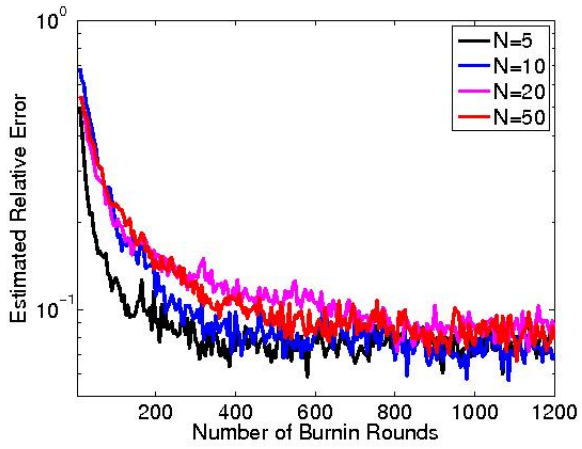


Figure 5: Convergence of relative error in statistics of first five components in networks of various sizes. Errors are computed with respect to statistics that are generated with $N = 10,000$ rounds.

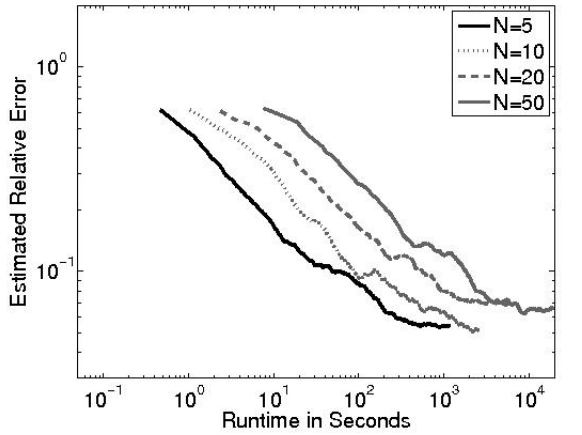


Figure 6: Relative error versus run-time in seconds for various network sizes.

with the size of the network.

While for experimental purposes we generate many samples independently. A practical strategy is to run a small number of chains in parallel and then collect take a large number of samples from each. We tested this strategy by generating 10 independent chain for various networks and estimating statistics from all samples except the first 20%. Using these, we measured how the behavior of error versus CPU run-time scales with network size. Average results of 9 independent tests are shown in Figure 6. Roughly, the run-time required for a certain level of accuracy scales linearly with network size.

Our sampling procedure is such that the cost of sampling a component depends on the time scales of its Markov neighbors and its own rate matrix. To demonstrate that, we

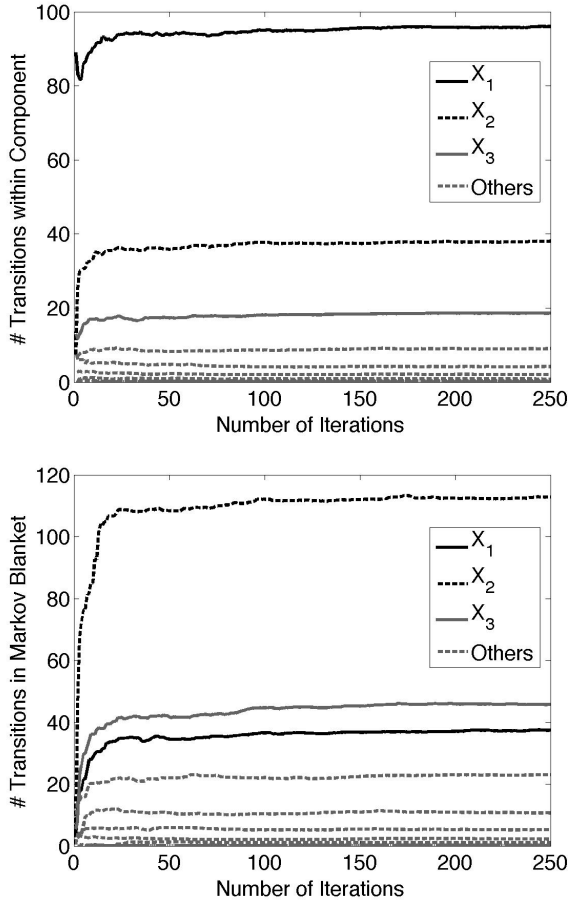


Figure 7: The effect of different time scales on the sampling. In this network X_i's rate is twice as fast than X_{i+1}'s rate. (top) The number transitions sampled for each of the first four components as a function of iteration number. (bottom) The number of intervals of Markov neighbors of each component as a function of iteration number.

created a chain network where each component has rates that are of half the magnitude of its parent. This means that the first component tends to switch state twice as fast as the second, the second is twice as fast as the third, and so on. When we examine the number of transitions in the sampled trajectories Figure 7, we see that indeed they are consistent with these rates, and quickly converge to the expected number, since in this example the evidence is relatively weak. When we examine the number of intervals in the Markov blanket of each components, again we see that neighbors of fast components have more intervals. In this graph X_1 is an anomaly since it does not have a parent.

6 Discussion

In this paper we presented a new approach for approximate inference in Continuous-Time Bayesian Networks. By building on the strategy of Gibbs sampling. The core

of our method is a new procedure for exact sampling of a trajectory of a single component, given evidence on its end points and the full trajectories of its Markov blanket components. This sampling procedure adapts in a natural way to the time scale of the component, and is exact, up to a predefined resolution, without sacrificing efficiency.

This is the first MCMC sampling procedure for this type of models. As such it provides an approach that can sample from the exact posterior, even for unlikely evidence. As the current portfolio of inference procedures for continuous-time processes is very small, our procedure provides another important tool for addressing these models. In particular, since the approach is *asymptotically unbiased* in the number of iterations it can be used to judge the systematic bias introduced by other, potentially faster, approximate inference methodologies, such as the one of Saria et al. (2007).

It is clear that sampling complete trajectories is not useful in situations where we expect a very large number of transitions in the relevant time periods. However, in many applications of interest, and in particular our long term goal of modeling sequence evolution (El-Hay et al., 2006), this is not the case. When one or few components transitions much faster than neighboring components, then we are essentially interested in its average behavior (Friedman and Kupferman, 2006). In such situations, it would be useful to develop a Rao-Blackwellized sampler that integrates over the fast components.

As with many MCMC procedures, one of the main concerns is the mixing time of the sampler. An important direction for future research is the examination of methods for accelerating the mixing - such as *Metropolis-coupled MCMC* or *simulated tempering* (Gilks et al., 1996) - as well as a better theoretic understanding of the convergence properties.

Acknowledgments

We thank Ido Cohn and the anonymous reviewers for helpful remarks on previous versions of the manuscript. This research was supported in part by grants from the Israel Science Foundation and the US-Israel Binational Science Foundation. Tal El-Hay is supported by the Eshkol fellowship from the Israeli Ministry of Science.

References

Chung, K. (1960). *Markov chains with stationary transition probabilities*. Springer Verlag, Berlin.

El-Hay, T., Friedman, N., Koller, D., and Kupferman, R. (2006). Continuous time markov networks. In *Proceedings of the Twenty-second Conference on Uncertainty in AI (UAI)*.

Fan, Y. and Shelton, C. (2008). Sampling for approximate inference in continuous time Bayesian networks. In *Tenth International Symposium on Artificial Intelligence and Mathematics*.

Friedman, N. and Kupferman, R. (2006). Dimension reduction in singularly perturbed continuous-time Bayesian networks. In *Proceedings of the Twenty-second Conference on Uncertainty in AI (UAI)*.

Gardiner, C. (2004). *Handbook of stochastic methods*. Springer-Verlag, New-York, third edition.

Gikhman, I. and Skorokhod, A. (1975). *The theory of Stochastic processes II*. Springer Verlag, Berlin.

Gilks, W. R., S., R., and J., S. D. (1996). *Markov Chain Monte Carlo in Practice*. Chapman & Hall.

Heskes, T. and Zoeter, O. (2002). Expectation propagation for approximate inference in dynamic Bayesian networks. In *Uncertainty in Artificial Intelligence: Proceedings of the Eighteenth Conference (UAI-2002)*, pages 216–233.

Minka, T. P. (2001). Expectation propagation for approximate Bayesian inference. In *Proc. Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI '01)*, pages 362–369.

Ng, B., Pfeffer, A., and Dearden, R. (2005). Continuous time particle filtering. In *Proceedings of the 19th International Joint Conference on AI*.

Nodelman, U., Shelton, C., and Koller, D. (2002). Continuous time Bayesian networks. In *Proc. Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI '02)*, pages 378–387.

Nodelman, U., Shelton, C., and Koller, D. (2003). Learning continuous time Bayesian networks. In *Proc. Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI '03)*, pages 451–458.

Nodelman, U., Shelton, C., and Koller, D. (2005). Expectation propagation for continuous time Bayesian networks. In *Proc. Twenty-first Conference on Uncertainty in Artificial Intelligence (UAI '05)*, pages 431–440.

Saria, S., Nodelman, U., and Koller, D. (2007). Reasoning at the right time granularity. In *Proceedings of the Twenty-third Conference on Uncertainty in AI (UAI)*.

Convex Point Estimation using Undirected Bayesian Transfer Hierarchies

Gal Elidan
Computer Science Dept.
Stanford University
Stanford, CA 94305
galel@cs.stanford.edu

Ben Packer
Computer Science Dept.
Stanford University
Stanford, CA 94305
bpacker@cs.stanford.edu

Geremy Heitz
Computer Science Dept.
Stanford University
Stanford, CA 94305
gaheitz@cs.stanford.edu

Daphne Koller
Computer Science Dept.
Stanford University
Stanford, CA 94305
koller@cs.stanford.edu

Abstract

When related learning tasks are naturally arranged in a hierarchy, an appealing approach for coping with scarcity of instances is that of transfer learning using a hierarchical Bayes framework. As fully Bayesian computations can be difficult and computationally demanding, it is often desirable to use posterior point estimates that facilitate (relatively) efficient prediction. However, the hierarchical Bayes framework does not always lend itself naturally to this maximum a-posteriori goal. In this work we propose an undirected reformulation of hierarchical Bayes that relies on priors in the form of similarity measures. We introduce the notion of "degree of transfer" weights on components of these similarity measures, and show how they can be automatically learned within a joint probabilistic framework. Importantly, our reformulation results in a convex objective for many learning problems, thus facilitating optimal posterior point estimation using standard optimization techniques. In addition, we no longer require proper priors, allowing for flexible and straightforward specification of joint distributions over transfer hierarchies. We show that our framework is effective for learning models that are part of transfer hierarchies for two real-life tasks: object shape modeling using Gaussian density estimation and document classification.

1 Introduction

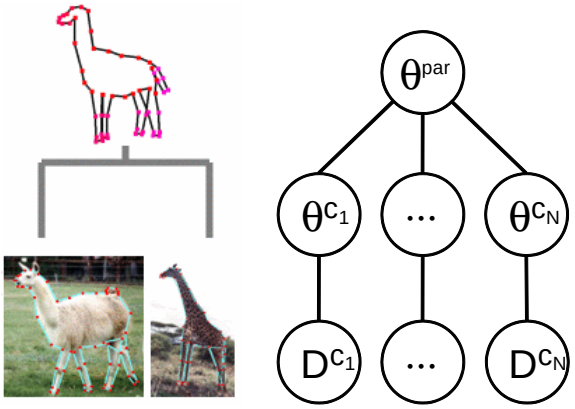
Many learning algorithms estimate a parametric model that is intended to generalize well over test datasets. The robustness and success of such methods rely, in large part, on the availability of sufficiently many training instances. In many applications, however, the availability of data is limited and alternative sources of information must be used.

An appealing approach for coping with this scenario involves what is called *transfer learning* (Thrun, 1996; Caruna, 1997), in which data from "similar" tasks/distributions is used to compensate for the sparsity of training data in our primary class or task. When learning a model for the shape of a giraffe, for example, we would like to take advantage of available instances of llamas; when learning to predict the topic of a document, we would like to take advantage of available documents of similar topics. In this paper, we consider the problem of transfer learning in the context of both density estimation and classification.

One general purpose approach for transfer learning is the simple *shrinkage* method designed to improve the estimates of ill-defined problems (e.g., (Carlin and Louis, 1996; McCallum et al., 1998)). This approach involves first learning the parameters of each task/distribution independently and then smoothing the parameters of the most fine-grained tasks (e.g., giraffe, deer) toward the parameters of the coarser-grained tasks (e.g., quadruped).

A more principled approach for transfer learning is the hierarchical Bayes framework, where similar models are related via an existing but unknown prior distribution over the common parameters, and Bayes rule is used to compute posterior parameter estimates (see, for example, Gelman et al. (1995)). This approach is conceptually elegant and effective in many settings, allowing the individual parameters to conform more or less to the prior belief over their values defined by the "parent" task/distribution. For example, it is natural to informally think of the shape distribution of a quadruped as a prior for the shape distribution of a giraffe or a deer. It can also be shown that, asymptotically, using the hierarchical Bayes approach is better than learning related distributions independently, particularly when the prior classes have complex structure (Baxter, 1997).

In practice, as full Bayesian computations can be both difficult and computationally demanding, it is often desirable to perform point estimation of the maximum a-posteriori (MAP) parameters, and use these parameters for (relatively) efficient prediction. Hierarchical Bayes techniques, however, have been designed primarily with Bayesian com-



(a) (b)

Figure 1: (a) Illustration of a simple transfer hierarchy where the distributions over the shapes of a pair of mammals share a common prior distribution (e.g., long-necked mammal). (b) An example of a hierarchical Bayes parametrization. The joint distribution over the data (observations of the leaf classes) and parameters has the form of $P(\Theta_{par}) \prod_i P(\Theta^{C_i} \mid \Theta_{par}) P(D^{C_i} \mid \Theta^{C_i})$.

putation in mind and are often not geared toward efficient point estimation. In particular, since many common priors such as the Dirichlet or normal-inverse-Wishart are not convex with respect to their arguments, the MAP objective does not lend itself naturally to efficient and optimal optimization. In addition, due to mathematical technicalities, the parametrization of such conjugate priors is often problematic for high-dimensional domains (see Section 2).

In this work we aim to preserve the probabilistic appeal of the hierarchical Bayes setting while applying it to the problem of posterior point estimation in high-dimensional transfer hierarchies. As in the standard hierarchical Bayes approach, we define a single probabilistic objective that combines fitting the data with terms that encourage soft similarity between several related distributions and a common parent distribution (see Figure 1). Unlike the standard hierarchical Bayes approach, we replace the prior distributions with positive similarity measures and write our joint probability distribution as an undirected Markov random field (MRF). This can be viewed as defining an improper prior over the parameters of all classes. This model falls under the broad definition of probabilistic abstraction hierarchies (PAHs) of Segal et al. (2001), but refines this general framework by defining a coherent joint distribution over all parameters in the hierarchy.

The hierarchical Bayes perspective also motivates an important extension of the joint distribution that is not captured by a standard shrinkage approach (e.g., (McCallum et al., 1998)) or by the PAH model (Segal et al., 2001). We introduce the notion of "degrees of transfer" along the edges of the hierarchy, allowing for different weights on different components of the similarity measures. These weights afford the possibility of more intelligent sharing in that similarity need not be equally strong throughout the hierarchy. In Section 4, we show how this idea is made concrete and how the weights can be automatically learned using the same optimization-based approach.

The reformulation of a hierarchical Bayes objective as an undirected model with improper priors has several benefits. First, specifying the joint distribution is straightforward, as it only requires a choice of a divergence measure between the parameters of the parent and child distributions. Second, multi-tier hierarchies present no additional complications, as conjugacy is not required for the goal of point estimation. Most importantly, if we choose a convex divergence term (e.g., L1, L2, e-insensitive, Kullback-Leibler), the problem of finding posterior point estimates is convex for many learning scenarios and can be solved using standard gradient based techniques.

We make our approach concrete for two very different settings. First, we consider a continuous distribution setting in which the shapes of different mammals share a common quadruped shape prior. Second, we construct a multi-tier hierarchy of naive Bayes models for document classification into topics based on the newsgroup dataset. In both cases we demonstrate that our high-dimensional undirected hierarchical Bayes model, joined with convex point estimation of posterior parameters, is superior to distributions learned independently as well as to the shrinkage approach.

2 Generative Hierarchical Bayes

Our goal is to define a framework for flexible transfer learning in a class hierarchy. In this section, we briefly describe the standard hierarchical Bayes approach for this scenario. In the next section, we present our alternative formulation.

We begin with a given hierarchy over a set of related learning tasks/classes $\mathcal{C}$ (for example, Figure 1(b)). At the lowest level are the leaf classes $c \in \mathcal{L}$ for which we observe instances $\mathcal{D}^c$. For convenience, we assume that $\mathcal{D}^c$ only exists for the nodes at the leaves of the hierarchy. [1] For each class c (except the root), $par(c)$ denotes its parent class.

The hierarchical Bayesian framework views this hierarchy as a directed probabilistic model in which the class parameters Θ^c are generated according to the distribution $P(\Theta^c \mid \Theta^{par(c)})$ and the data is generated according to $P(\mathcal{D}^c \mid \Theta^c)$ for $c \in \mathcal{L}$. This induces a joint distribution over the observed data and all class parameters as follows:

$$P(\mathcal{D}, \Theta) = \prod_{c \in \mathcal{L}} P(\mathcal{D}^c \mid \Theta^c) \times \prod_{c \in \mathcal{C}} P(\Theta^c \mid \Theta^{par(c)}) \quad (1)$$

where for the root class $P(\Theta^c \mid \Theta^{par(c)}) \equiv P(\Theta^c)$.

[1] Our general formulation in Section 3 can easily incorporate observations of any class in the hierarchy.

Once this directed model is defined, we are typically interested in the posterior distribution of the parameters $P(\Theta \mid \mathcal{D})$, or the marginal of this posterior for a subset of the parameters. In the *empirical Bayes* approach, tailored for the common case of a two-level hierarchy, a point estimate for the root parameters is found first, and then full posteriors $P(\Theta^c \mid \mathcal{D}, \Theta^{root})$ are computed for the leaf distributions (see Gelman et al. (1995) for more details).

As noted in Section 1, it is often preferable to compute point estimates of the MAP parameters rather than use full posteriors for the practical purpose of prediction. That is, we want to compute

$$\Theta^* = \operatorname*{argmin}_{\Theta} - \log P(\mathcal{D}, \Theta),$$

where

$$\begin{aligned} \log P(\mathcal{D}, \Theta) \quad = \quad & \sum_{c \in \mathcal{L}} \log P(\mathcal{D}^c \mid \Theta^c) \\ & + \sum_{c \in \mathcal{C}} \log P(\Theta^c \mid \Theta^{par(c)}) \end{aligned} \tag{2}$$

By design, many standard hierarchical Bayes approaches lend themselves toward the fully Bayesian treatment, and therefore make limiting choices. In particular, priors are generally chosen to be in conjugate families to allow for the computation of posteriors in closed form. However, many common conjugate priors such as the Dirichlet or normal-inverse-Wishart are not convex with respect to the parameters, thus not allowing for efficient point estimation.

In addition, the requirement that the priors be valid distributions that integrate to 1 can lead to untenable restrictions. For example, the conjugate prior for the multivariate Gaussian is the normal-inverse-Wishart distribution, which has a degrees of freedom parameter corresponding to the strength (pseudocounts) of the prior. In order to ensure that it is a proper distribution, the pseudocounts must be at least as large as the dimension d of the data. Thus, in the case of high-dimensional data with few instances, this requirement forces the strength of the prior to overwhelm the signal in the data. For example, in Section 5 we consider 120-dimensional data representing the shape of various mammals, with up to only 15 instances per class. Even if we perform dimensionality reduction so that the data is only 20-dimensional, using a normal-inverse-Wishart in this case would mean that the weakest allowable prior would be at least as strong as the data. Indeed, most hierarchical Bayes applications are limited to the scenario in which the higher level priors are low-dimensional and relatively simple (e.g., (Gelman et al., 1995; Blei et al., 2003)), rather than a rich class distributions in their own right.

3 Undirected Transfer Hierarchies

As mentioned above, many of the difficulties associated with the standard hierarchical Bayes approach result from the design of the framework for full posterior estimation. Thus, for the purpose of point posterior estimation, we propose an alternative formulation that is more amenable to the use of standard optimization techniques. We start by describing the general approach and then provide two different scenarios for which it is made concrete. In Section 4 we present an extension to the basic approach that allows for greater flexibility in the way that parameters are shared.

3.1 Basic Framework

We begin our reformulation by writing Eq. (2) as an objective that has a more general form (and that can be made equivalent to Eq. (2) given the appropriate instantiation):

$$\begin{aligned} F_{\text{joint}}(\Theta; \mathcal{D}) = & - \sum_{c \in \mathcal{L}} \mathcal{F}_{\text{data}}(\mathcal{D}^c, \Theta^c) \\ & + \beta \sum_{c \in \mathcal{C}} \text{Div}(\Theta^c, \Theta^{par(c)}) \end{aligned} \tag{3}$$

Here, $\mathcal{F}_{\text{data}}(\mathcal{D}^c, \Theta^c)$ is a data-dependent objective (e.g., the probability of $\mathcal{D}^c$ given Θ^c) that encourages the parameters of each class to represent the data well. $\text{Div}(\Theta^c, \Theta^{par(c)})$ is a divergence or dissimilarity function over the child and parent parameters that encourages the parameters of linked classes to be similar. The weight β determines the overall trade-off between these competing objectives.

In the analysis that follows, we consider divergence functions that decompose independently over each parameter in Θ^c. This includes many dissimilarity metrics, including all norms. Using this assumption, we write:

$$\begin{aligned} F_{\text{joint}}(\Theta; \mathcal{D}) = & - \sum_{c \in \mathcal{L}} \mathcal{F}_{\text{data}}(\mathcal{D}^c, \Theta^c) \\ & + \beta \sum_{c \in \mathcal{C}} \sum_{i} \text{Div}(\theta_i^c, \theta_i^{par(c)}) \end{aligned} \tag{4}$$

Note that we can more generally take Θ_i^c to be (possibly overlapping) subsets of Θ^c. For the sake of simplicity, however, we only consider the fully decomposed form.

There are several points to note about our objective in Eq. (4). First, it has the general form of the log-probability of a *Markov random field* (MRF) (Pearl, 1988), and if its integral is finite, then $\frac{1}{Z} \exp\{-F_{\text{joint}}(\Theta; \mathcal{D})\}$ defines a coherent joint probability function over the data and parameters of all classes in the hierarchy. Thus, this formulation is in fact a hierarchical Bayes setting where we make use of unnormalized priors in the form of $\exp\{-\text{Div}(\theta_i^c, \theta_i^{par(c)})\}$. Second, the penalty term is a function of both the child and parent parameters. This means that the penalty can be reduced by either parameter moving towards the other. It is this feature that differentiates us from empirical Bayes.

While Eq. (3) and Eq. (4) may appear as deceptively naive rewritings of Eq. (2), this view actually has important practical ramifications. The first benefit is the ease with which

relatedness may be defined. Natural choices include the L2 distance (corresponding to a local unnormalized Gaussian prior), the L1 distance (corresponding to a local unnormalized Laplacian prior), or a (smoothed) ϵ-insensitive loss (which gives a constant penalty for all values below ϵ).

The second benefit is that we can now use inference/optimization techniques that have been developed for energy objectives that take on the form of Eq. (3). Our goal is to learn a point estimate for the parameters that is the mode of the posterior distribution. That is, given $\mathcal{D}^c$ for all classes, we want to learn the parameters Θ^* such that

$$\Theta^* = \operatorname*{argmin}_{\theta} F_{\text{joint}}(\Theta; \mathcal{D})$$

Appealingly, for *any* convex divergence function $\text{Div}(\Theta^c, \Theta^{par(c)})$, this entire objective is convex in Θ, when the data objective $\mathcal{F}_{\text{data}}(\mathcal{D}^c, \Theta^c)$ (e.g., likelihood) is concave, as is typically the case for many learning scenarios. This allows us to find the mode of optimal parameters using straightforward gradient ascent techniques; in the experiments below, we use the Polak-Ribiere conjugate gradient algorithm (Boyd and Vandenberghe, 2004). Note that our approach scales well with large amounts of data, as the sample size affects only the one-time collection of sufficient statistics for use in $\mathcal{F}_{\text{data}}$.

Below, we describe how the reformulation of the hierarchical Bayes model is made concrete for two different transfer learning scenarios. In Section 5 we present experiments using these instantiations for two real-life scenarios.

3.2 Gaussian Density Estimation

Suppose we wish to model the data of class c using a multivariate Gaussian distribution parametrized by mean μ^c and covariance Σ^c. The data objective for this class is naturally given by the (regularized) log-likelihood:

$$\begin{aligned} \mathcal{F}_{\text{data}}(\mathcal{D}^c, \Theta^c) &\equiv \ell\left(\mathcal{D}^c; \Theta^c\right) \\ &= \sum_{m}^{M_c} \log \mathcal{N}(\mathbf{x}[m] \mid \mu^c, \Sigma^c + \alpha\mathcal{I}), \end{aligned} \tag{5}$$

where $\mathbf{x}[m]$ is the mth instance in $\mathcal{D}^c$ (which is of size M_c), $\mathcal{I}$ is the identity matrix, and α is the standard regularizing "ridge" term used to avoid issues of singular matrices and overfitting in the regime of small amounts of data.

This objective is not concave in the covariance parameter Σ^c, but becomes concave if the Gaussian is parametrized using the *inverse* covariance matrix, also known as the precision matrix. In this case, the log-likelihood is:

$$\begin{aligned} \ell\left(\mathcal{D}^c; \Theta^c\right) = &-\frac{1}{2}\sum_{m}^{M_c}(\mathbf{x}[m]-\mu)^T K(\mathbf{x}[m]-\mu) \\ &- \frac{M}{2}\log\det K + C \end{aligned} \tag{6}$$

where $K^{-1} = \Sigma^c + \alpha\mathcal{I}$.

Unfortunately, the Gaussian log-likelihood is *not* jointly concave in the mean vector and precision matrix. However, it *is* concave in each part independently. As a result, we can iteratively optimize our objective in two phases, one for each set of parameters. This process generally converges to a local optimum in a few iterations.

For the divergence function, we use the L2 norm over the mean and diagonal precision parameters, while we do not tie the off-diagonal terms to the parent parameters.

3.3 Discrete Bayesian Networks

In contrast to the case of continuous Gaussian distributions, we consider the setting in which each class model is a Bayesian network over discrete-valued data. Note that in this scenario, each node in the hierarchy is a Bayesian network, and we assume that the networks have the same structure and therefore the same parametrization Θ^c.

We consider the specific case of table CPDs (conditional probability distributions) using a multinomial distribution. To avoid positivity and normalization constraints, we use a log-space representation of the multinomial distribution

$$P(x \mid pa_X; \theta^c) = \frac{\exp(\theta^c_{x,pa_X})}{\sum_{x'}\exp(\theta^c_{x',pa_X})},$$

where x and pa_X are specific values for the variable X and its parents. Note that the notion of the parents of X is in the context of the Bayesian network that sits inside each node in the hierarchy; this is distinct from the notion of parents of the nodes themselves in the hierarchy.

We now define the data objective to be the log-likelihood of the data of a class c given the parameters Θ^c of the corresponding Bayesian network:

$$\begin{aligned} \mathcal{F}_{\text{data}}(\mathcal{D}^c, \Theta^c) &\equiv \ell\left(\mathcal{D}^c; \Theta^c\right) \\ &= \sum_{m}^{M_c}\sum_{X\in\mathcal{X}} \log P(x[m] \mid pa_X[m]; \theta^c) \\ &= \sum_{X\in\mathcal{X}}\sum_{x}(N_c\{x, pa_X\} + \alpha)\cdot \\ &\qquad \{\theta^c_{x,pa_X} - \log\sum_{x'}\exp(\theta^c_{x',pa_X})\} \end{aligned} \tag{7}$$

where the sufficient statistic $N_c\{x, pa_X\}$ is the number of times the values (x, pa_X) occur in the dataset $\mathcal{D}^c$, and α is the corresponding pseudo-count of the Dirichlet prior distribution used to regularize the likelihood. For the divergence function, we use the L2 norm over all parameters θ^c.

4 Degree of Transfer Coefficients

Our model thus far can be viewed as a probabilistic reinterpretation of the PAH framework of Segal et al. (2001).

Having motivated our model as an undirected reformulation of the hierarchical Bayes joint distribution, we can use this novel perspective to suggest additional modifications that are natural to this setting. In particular, we can refine the single weight β that corresponds to prior strength in Eq. (4) and introduce distinct $\lambda_i^{c,par(c)}$ terms at each edge $(c, par(c))$ in the hierarchy. These *degree of transfer* (DOT) coefficients represent how much we want each of the child parameters to be near its corresponding parent parameter. Note that, as with the $\text{Div}(\Theta^c, \Theta^{par(c)})$ functions, we can define DOT coefficients for groups of parameters rather than individual ones.

To make the benefit of different degrees of transfer coefficients concrete, consider a document classification task as a motivating example. Suppose that the child class is the 'biology textbook' topic with a small number of training instances and the parent class is the 'science textbook' topic, with a significantly larger number of training instances. The frequency of the word 'experiment', represented by the multinomial parameter $\theta_{\text{experiment}}$, is likely to be similar for both the child and parent class. On the other hand, the parameter θ_{gene} corresponding to the word 'gene' is likely to be misrepresented at the parent class. We therefore stand to gain from penalizing the distance between child and parent more for $\theta_{\text{experiment}}$ and less for θ_{gene}.

Incorporating the degree of transfer coefficients $\lambda_i^{c,par(c)}$, our objective now becomes:

$$F_{\text{joint}}(\Theta, \Lambda; \mathcal{D}) = -\sum_{c \in \mathcal{L}} \mathcal{F}_{\text{data}}(\mathcal{D}^c, \Theta^c) + \beta \sum_{c \in \mathcal{C}} \sum_i \frac{1}{\lambda_i^{c,par(c)}} \text{Div}(\theta_i^c, \theta_i^{par(c)}) \quad (8)$$

This allows us the flexibility to turn sharing on and off in a continuous fashion for the various edges in the hierarchy. For instance, if we set $\lambda_i^{c,par(c)}$ near zero, we are effectively forcing the parameters to agree, whereas if we make it large, we are allowing as much flexibility as is necessitated by the likelihood term.

Naively, we might try treating these factors as parameters of the joint transfer objective of Eq. (8) and optimize with respect to both the class parameters and transfer factors. Upon examination of the objective in Eq. (8), however, we notice that the optimal value occurs when all transfer parameters approach infinity ($\lambda_i^{c,par(c)} \to \infty$) and all model parameters are set to their independent maximum likelihood values. This should not come as a surprise, as the transfer parameters play the role of a prior strength, which is usually estimated using cross validation or some other external means. Obviously, while such an approach can be used to estimate the global weight β, we cannot hope to cross validate the large number of λ parameters. How then can we estimate the parameter values in a meaningful way?

Our first option is to choose a reasonable *constant* value for $\lambda_i^{c,par(c)}$ before we optimize the parameters Θ. We describe an empirical Bayes bootstrap approach for doing so in Section 4.1. In Section 4.2, we describe a different approach that places a prior over the λ coefficients and optimizes them along with the model parameters Θ.

4.1 Undirected Empirical Bayes Estimation

In order to choose an appropriate constant value for each $\lambda_i^{c,par(c)}$, recall that it represents the inverse strength of the divergence penalty between a child and parent parameter. Thus, it should be assigned a lower value if we expect the parameter to be near its parent, and a higher value if we expect it to be far. One way to quantify this is to consider randomly sampled subsets of the child data $\mathcal{D}^c$: if the maximum likelihood estimate for a parameter θ_i^c is consistently close to the corresponding parent parameter $\theta_i^{par(c)}$ across the sampled datasets, then we want to encourage higher similarity; otherwise, we set a lower penalty coefficient.

Formally, we define a random variable $\delta_{i,c} = \theta_i^c - \theta_i^{par(c)}$ and wish to estimate the expected variance of this variable *across all possible datasets*. We use the bootstrap approach of Efron and Tibshirani (1993) to approximate this expectation. That is, for each class, we create K random datasets of size equal to the original training set by uniformly sampling with replacement from the original dataset. For each of these sets we then estimate the standard (regularized) maximum likelihood parameters θ^c and $\theta^{par(c)}$, and compute the empirical variance of the difference $\delta_{i,c}$ across the K trials, which we denote by $\hat{\sigma}^2_{c,par(c),i}$. We then use this disparity measure to set the transfer coefficients:

$$\lambda_i^{c,par(c)} = \hat{\sigma}^2_{c,par(c),i}$$

Using this approach together with, for example, the quadratic penalty, our objective becomes

$$F_{\text{joint}}(\Theta; \mathcal{D}, \Lambda) = -\sum_{c \in \mathcal{L}} \mathcal{F}_{\text{data}}(\mathcal{D}^c, \Theta^c) + \beta \sum_{c \in \mathcal{C}} \sum_i \frac{(\theta_i^c - \theta_i^{par(c)})^2}{\hat{\sigma}^2_{c,par(c),i}}. \quad (9)$$

Under this formulation, we see that the penalty terms in the objective represent a product of Gaussian priors over the difference between the parent and child values of parameter θ_i. If we were to fix the parent parameter to its bootstrap value, this would reduce to empirical Bayes using the parent bootstrap estimates as parameters of the Gaussian prior. We therefore call this approach the undirected empirical Bayes estimation of the transfer factors.

The approach proposed in this section is appealing from a computational perspective for two reasons. First, the complexity of the estimation of the transfer factors is proportional to the complexity of the estimation of the ML parameters of each class independently. Second, given the

transfer factors that are computed only once, the objective Eq. (9) is convex for many typical learning tasks and choices of the divergence function $\text{Div}(\Theta^c, \Theta^{par(c)})$ and can thus be efficiently optimized.

4.2 Hyperprior-Based Estimation

A second approach is to estimate each DOT transfer coefficient along with the model parameters themselves. To ensure that these DOT coefficients are not driven to infinity, we can add a prior that pushes the factors towards smaller values. Specifically, we can add an inverse-Gamma prior to the transfer factors, forcing them to be positive. This is a natural choice since the inverse-Gamma is the conjugate prior distribution for Gaussian variances, and we can (loosely) interpret λ as the variance of a Gaussian prior for the model parameter. In this case, our objective expands to:

$$\begin{aligned} F_{\text{joint}}(\Theta, \Lambda; \mathcal{D}) = &\sum_c -\ell\left(\mathcal{D}^c; \Theta^c\right) \\ &+ \beta \sum_{c \in \mathcal{L}} \sum_i \frac{(\theta_i^c - \theta_i^{par(c)})^2}{\lambda_i^{c,par(c)}} \\ &- \sum_{c \in \mathcal{C}} \sum_i \log G^{-1}(\lambda_i^{c,par(c)}) \end{aligned} \quad (10)$$

where G^{-1} is the density function of the inverse-Gamma distribution. We choose the parameters of the inverse-Gamma distribution such that the mean is equal to the bootstrapped value $\hat{\sigma}^2_{c,par(c),i}$ as in Section 4.1.

Importantly, if our data objective (e.g., log-likelihood) function is concave, our entire objective is jointly convex in the model parameters (θ's) and the DOT coefficients (λ's). This is an attractive aspect of our formulation, and allows for efficient and guaranteed optimal inference of the MAP values for the parameters.

5 Experimental Evaluation

In this section, we demonstrate how our reformulation of the hierarchical Bayes model as a Markov random field allows us to effectively transfer across classes. We consider the task of density estimation for multivariate Gaussian shape models as well as a document classification task. In all experiments we compare our hierarchical approach to a **CV Reg** model for which each class is learned independently. The parameters of **CV Reg** are chosen to maximize the regularized likelihood, and the regularization coefficients are determined using cross-validation. To provide an additional baseline, we also compare to a **Shrinkage** approach (see below for the details in each application).

5.1 Mammal Shape Model Learning

We adopt the notion of object shape modeled by a Gaussian distribution as in Elidan et al. (2006) and attempt to learn a shape model for several classes of mammals. Each instance is a set of 60 landmarks from hand-outlined images represented as a 120-dimensional vector (using the x and y coordinates of each landmark). Our dataset consists of 40 such instances of Elephants, Bison, Rhinos, Giraffes, and Llamas. Since these classes are well-represented on a lower-dimensional manifold, we use PCA to reduce the dimensionality to 20. The objective formulation and covariance matrix regularization are explained in Section 3.2.

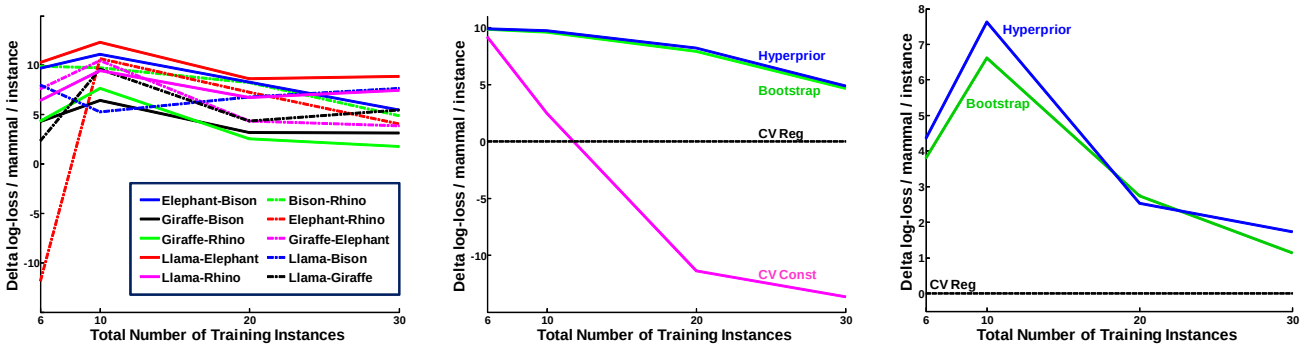
We evaluate the benefit of transferring between pairs of mammal classes. We train each method using $N = \{3, 5, 10, 15\}$ instances for each child class, evaluating the log-likelihood of 20 test instances for each child class. We report averages using five fold cross-validation.

To demonstrate the usefulness of the degree of transfer (DOT) coefficients, we consider three variants of our method. **Bootstrap** uses bootstrapping to estimate the DOT coefficients as described in Section 4.1. **Hyperprior** uses the approach described in Section 4.2 to define a hyperprior distribution over the DOT coefficients and estimate the DOT coefficients together with the other model parameters. In both these cases we set the global weight β to 1. To evaluate whether including the DOT coefficients provides any advantage, the **CV Const** method does not use DOT coefficients, instead cross-validating β in the range of 10^{-6} to 1 (a typical value range of the coefficient when using **Bootstrap**). To make it as competitive as possible, we cross-validate specifically for each mammal pair and for each number of training instances N.

We compare these variants of our method to the independent **CV Reg** baseline, which uses cross-validated regularization as described in Section 3.2. Our experiments showed that a standard **Shrinkage** approach performs very poorly in this case due to the fact that linearly interpolating covariance matrices yields nonsensical (and possibly invalid) distributions. We attempted a version of shrinkage over the diagonal entries of the covariance alone, but this also resulted in poor performance, as the appropriate interaction between the variance (diagonal) and covariance (off-diagonal) elements is disturbed. We therefore omit the **Shrinkage** variant from the graphs reported below.

Figure 2 shows the difference in test log-likelihood between our methods and the **CV Reg** baseline. (a) shows the benefit of our **Hyperprior** approach for all mammal pairs. The benefit of our approach is evident and typically lies in the range of $5-10$ bits per instance on test data. Out of 10 mammal pairs, our method is not superior to **CV Reg** in only a single case and only when the number of training instances of each class is 3.

In Figure 2(b) we consider the rhino-bison mammal pair, which is "close" in shape, and compare the success of the different variants of our method. The benefit of having independent DOT coefficients is clear as **CV Const** is in-



(a) (b) (c)

Figure 2: Results for transfer between mammal pairs. The numbers reported are the average across 5 randomized folds of the improvement in test log-likelihood per mammal per instance (y-axis) over the regularized **CV Reg** baseline, as a function of number of samples (x-axis). (a) shows average improvement in performance for all mammal pairs of our method with a **Hyperprior** over the degree of transfer (DOT) coefficients; (b) compares **Hyperprior** with our **Bootstrap** estimation of the DOT coefficient as well as the cross-validated **CV Const** for the "similar" bison and rhino pair; (c) is the same as (b) for the "dissimilar" llama and rhino pair. In (c), the **CV Const** method is omitted, as it is at least 100 bits per instance worse than **CV Reg** for all training set sizes. Typically, **Hyperprior** training time was approximately 90 seconds.

ferior, due to the fact that samples are scarce and cross-validation is not robust. Figure 2(c) shows an example of the giraffe-rhino mammal pair that are farther apart. For this pair, **CV Const** was at least 100 bits per instance worse than **CV Reg** for all training set sizes, so it is omitted from the graph. Although the **Hyperprior** method is somewhat more advantageous than the **Bootstrap** approach, we note that overall the methods are quite competitive.

5.2 Newsgroup Posting Classification

In order to demonstrate our method on a larger and more involved hierarchy than the simple mammal pairs, we use the Newsgroup dataset, which consists of 18,827 newsgroup postings drawn from 20 distinct newsgroups. For the experiments described here, we use a hierarchy over 15 classes, gathered into the abstract classes Religion, Politics, Vehicles, Sports, and Computers, as in McCallum et al. (1998). This creates a hierarchy with 15 leaf nodes, five middle level node, and one root node, as shown in Figure 3(a).

Documents are tokenized and all tokens occurring only one time are removed to produce a corpus with 55,989 unique words. When individual experiments are performed, we use 100 test documents, and all words not present in the training or test set for that particular experiment are removed (to improve efficiently and clarity of the results), typically leaving around 20000 words. We report average results using 5 fold cross validation.

In addition to the regularized **CV Reg** model, we implemented a slightly simplified variant of hierarchical **Shrinkage** (McCallum et al., 1998) for which, starting from the root of the node, each node's parameters are shrunk toward its parent using a single parameter learned by k-fold cross validation. Due to the size of the optimization problem in this case, we use a simplified **Undirected HB** version of our approach that does not make use of the individual DOT coefficients and where we set the global weight β to 1.

For both the baselines as well as our method, we use a naive Bayes model as an instantiation of the approach described in Section 3.3. A document d in this framework is modeled as a bag of words, where d_i indicates the number of times that word i appears in the document. We model each class as a probability distribution over words, where each word in the document is considered independently:

$$P(d \mid c) = \prod_i P(w_i \mid c)^{d_i}.$$

We use a multinomial distribution as in Section 3.3 for $P(w_i \mid c)$. The full data objective for this model is:

$$\mathcal{F}_{\text{data}}(\mathcal{D}^c, \Theta^c) \equiv \sum_{d \in \mathcal{D}^c} \sum_i (d_i + \alpha) \cdot \{\theta_i^c - \log \sum_j \exp(\theta_j^c)\}$$

We evaluate each method using its classification rate on test instances. To highlight the importance of regularization (through the parameter α), we also present the baseline **Likelihood** that simply uses the maximum likelihood parameters for each class without regularizing. Figure 3 shows the consistent advantage of our **Undirected HB** approach over all baselines. The advantage of the regularized **CV Reg** over **Likelihood** is also clear, demonstrating the extent to which even simple regularization, once cross-validated, can be beneficial. This also explains why the regularized **CV Reg** model achieves a better classification rate than the **Shrinkage** method that uses Laplacian smoothing (setting $\alpha = 1$), as in McCallum et al. (1998).

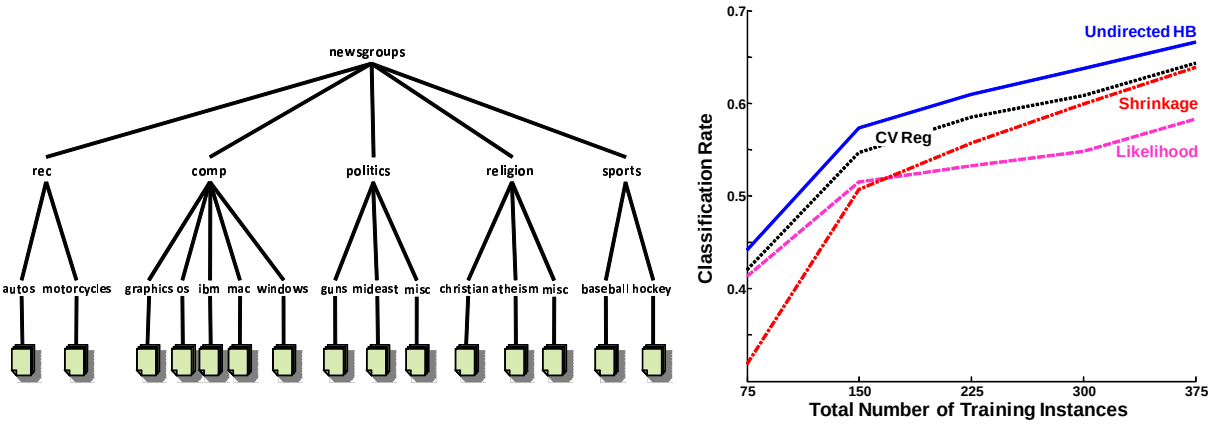


Figure 3: (left) A hierarchy of topics for the Newsgroup dataset adopted from McCallum et al. (1998). (right) Results for the 15 class Newsgroup hierarchy. Shown is average classification accuracy across 5 random train and test splits (y-axis) as a function of number of samples (x-axis). Our **Undirected HB** approach is compared to the recursive **Shrinkage** approach, the regularized **CV Reg** model, and to non-regularized **Likelihood**. The typical **Hyperprior** training time was approximately 30 minutes, comparable to that of **Shrinkage**.

6 Summary and Future Directions

In this work, we proposed an undirected reformulation of the hierarchical Bayes framework that replaces proper priors with similarity measures for the purpose of posterior point estimation in high-dimensional transfer hierarchies. We introduced the notion of "degree of transfer" weights that afford our model flexibility not available in other standard hierarchical methods, and suggested two approaches for automatically estimating these weights. Finally, we showed how the general approach can be applied to the scenario of shape modeling using a multivariate Gaussian density and to document topic classification. In both settings, we demonstrated the superiority of our approach over both independent learning and a shrinkage-based competitor.

The benefits of our reformulation are threefold. First, while retaining the hierarchical Bayes appeal of using a prior as the mechanism for transfer, our method is straightforward and allows for straightforward specification of the objective. Second, by writing a Markov random field objective, we can use convex optimization techniques to find point estimates of the posterior distributions for a large range of learning scenarios and similarity measures. Third, our framework allows for different degrees of transfer for different parameters so that some parts of the distribution can be transferred to a greater extent than others.

There are several extensions to this work to explore. The first is learning the structure of transfer itself, at the class and parameter level. The PAH framework (Segal et al., 2001) allows the structure of the hierarchy to be learned; one could adapt their approach to both identify hierarchy edges and discover which parameters (e.g., those that correspond to a coherent part of a mammal) transfer together. Furthermore, unlike the classical hierarchical Bayes approach, there is nothing in our framework that prevents a class from having multiple parent classes. Such hierarchies are common in many settings (including the Wordnet hierarchy and the GO hierarchy in biology), and it would be interesting to explore whether this added flexibility can be of benefit in a Bayesian learning task.

References

J. Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28:7—39, 1997.

D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

S. Boyd and L. Vandenberghe. *Convex Optimization*. 2004.

B. Carlin and T. Louis. *Bayes and Empirical Bayes Methods for Data Analysis*. Chapman and Hall, 1996.

R. Caruna. Multitask learning. *Machine Learning*, 28:41–75, 1997.

B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

G. Elidan, G. Heitz, and D. Koller. Learning object shape: From cartoons to images. In CVPR, 2006.

A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman and Hall, 1995.

A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng. Improving text classification by shrinkage in a hierarchy of classes. In ICML, 1998.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

E. Segal, D. Koller, and D. Ormoneit. Probabilistic abstraction hierarchies. In NIPS, 2001.

S. Thrun. Is learning the n-th thing and easier than learning the first? In NIPS, 1996.

Learning and Solving Many-Player Games through a Cluster-Based Representation

Sevan G. Ficici
Harvard School of Engineering and Applied Sciences
sevan@eecs.harvard.edu

David C. Parkes
Harvard School of Engineering and Applied Sciences
parkes@eecs.harvard.edu

Avi Pfeffer
Harvard School of Engineering and Applied Sciences
avi@eecs.harvard.edu

Abstract

In addressing the challenge of exponential scaling with the number of agents we adopt a *cluster-based representation* to approximately solve asymmetric games of very many players. A cluster groups together agents with a similar "strategic view" of the game. We learn the clustered approximation from data consisting of strategy profiles and payoffs, which may be obtained from observations of play or access to a simulator. Using our clustering we construct a reduced "twins" game in which each cluster is associated with two players of the reduced game. This allows our representation to be *individually-responsive* because we align the interests of every *individual* agent with the strategy of its cluster. Our approach provides agents with higher payoffs and lower regret on average than model-free methods as well as previous cluster-based methods, and requires only few observations for learning to be successful. The "twins" approach is shown to be an important component of providing these low regret approximations.

1 Introduction

Consider the problem of solving non-cooperative games of realistic size. As the number of agents increases, the size of the game increases exponentially and it becomes intractable to even represent a game explicitly never mind solve for the equilibrium of the game. Recognizing this, one direction adopted in the literature on computational game theory is to assume that games have some underlying structure and focus on succinctly representable games; e.g., graphical games [6] and action-graph games [2]. But what if there is no exact, structured representation of a game? A second natural direction is to find a suitable approximation of the game, that can be solved and will provide a good model of the strategic characteristics of the actual game, so that agents have low regret from adopting the strategy determined by solving the approximate game. This is the direction we follow in the current paper.[1]

We consider the use of a *cluster-based representation*, in which the same strategy is ultimately prescribed to every agent in a cluster. A cluster groups together agents with a similar "strategic view" of the game. This means that they have similar payoffs and similar effects on other agents. We do not require that the actual game is symmetric and allow for agents with different payoff functions even within a cluster. We do not require explicit knowledge of the full game. Instead, we learn (offline) the cluster-based representation from data consisting of strategy profiles and payoffs. The data may be obtained from observations of play, or we may have access to a simulator with which to generate payoffs for different strategy profiles. We learn both the clustering and also the payoffs to agents in each cluster given each profile of cluster strategies.

A natural next step is to solve a clustered representation of the game to find a Nash equilibrium, and recommend the equilibrium strategies to agents in the original many-player game. However, using a naive clustered representation leads to a situation in which individual agents' interests differ from their clusters', and therefore individual agents will not adhere to the recommended strategies. To address this problem we construct a "twins" game in which every cluster is represented by a pair of players. This representation is *individually-responsive* because in addition to requiring that the cluster strategies form a Nash equilibrium of the reduced game induced by the clusters, we ensure that the interests of every *individual* agent within

[1] Finding approximation algorithms for solving games is an active area (see Daskalakis et al. [3] for a recent survey), but our focus is on effective, heuristic methods rather than on achieving worst-case approximation bounds.

a cluster are aligned with the cluster strategy. As a result, no individual agent within a cluster would like to deviate from the strategy prescribed to the cluster.

We present a motivating example of a vendor game in Section 2. In Section 3, we introduce our cluster-based model and explain our methods to learn a good clustering from observations of play (or access to a simulator) and to learn a good model of payoffs for a given clustering. In Section 4 we explain how the twins game is defined for a given instantiation of the cluster-based model, and explain why the representation is individually-responsive. We present experimental results in Section 5, where we consider the vendor game (where vendors can be complements or substitutes for each other) and a variant of the *Santa Fe bar problem*. We compare our cluster-based model with other approaches and show that our model provides solutions that give agents low regrets and good payoffs.

Related Work. Wellman et al. [11] propose and study an approximate representation that is suitable for symmetric games. To facilitate solving such a game, these authors group agents into multiple clusters, wherein all agents in each cluster are constrained to follow the same strategy. In our approach, we would use a single cluster for a symmetric game and solve it with the twins representation. Our twins approach provides individual responsiveness while allowing for multiple clusters, which is essential for asymmetric games. Furthermore, we do not require that the clustered representation be exact, or that the full game be known, but instead learn the cluster-based representation from observations.

On the other hand, Vorobeychik et al. [10] apply regression learning techniques to model payoffs for continuous games. While these authors consider some forms of strategy aggregation, they do not explicitly seek to combine learning with the use of reduced game-form representations. The novelty of our use of learning then is that we seek to integrate learning directly with our reduced game representation, both for the purpose of learning the structure of an appropriate cluster representation and also for learning the payoffs for the induced "twins" game. Further, our learning is applied to many player games, whereas they restrict attention to games with a small number of players.

The work on graphical games [6] assumes locality of interaction, while we make no such assumptions, and graphical games with many agents remain hard to solve. In comparing with action graph games [2], we note for example that our vendor game could be viewed as an action graph game *if the cluster-based approximation is in fact exact*. But solving the action graph game would still amount to solving a many player game because individual agents would need to take into account other agents in their own cluster. In contrast, our twins game allows us to reduce the number of players while still being individually responsive.

Daskalakis and Papadimitriou [4] consider anonymous games, a special case of action graph games in which the strategic considerations depend only on the *number* of agents that adopt each strategy but not their identity. They develop worst-case approximation results, including a polynomial time approximation scheme for finding an ϵ-approximate Nash equilibrium when the number of strategies is two.

Kearns and Mansour [7] present *summarization games*, a compact representation for games with many players. Their approach has two components: a *summarization function*, which maps the space of N-player strategy profiles onto the interval $[0, 1]$, and a set of *payoff functions*, one for each player. Each player's payoff function is a function of that player's strategy choice and the output of the summarization function. They establish conditions under which the Nash equilibria of summarization games can be approximated to within some epsilon in polynomial time, but do not consider the use of summarization games as *approximations* to other games. Indeed, whereas most work on approximation uses an exact representation of the game to obtain an approximate solution, we instead use an approximate representation of the game and obtain an exact solution to that approximation.

Jehiel [5] advances the idea of an *analogy-based expectation* equilibrium in which agents are clustered into "analogy classes" and each agent plays a best-response against the average strategy in each cluster. Whereas our approach solves a reduced twins game with two players per cluster, Jehiel's approach still maintains an explicit representation of every agent in solving for this equilibrium. Rather than computational tractability, Jehiel's focus is on providing prescriptive power for how people behave in strategic settings, along with alternate explanations of some well known paradoxes in extensive form games.

2 The Vendor Game

An example of a game that lends itself to our approach is a *vendor game*. Here, we have a large number of vendors, each selling a different product. The products belong to categories. For example, some vendors may sell drinks while others sell sandwiches. Within each category, the products are further differentiated. For example, one drinks vendor may sell beer while another sells lemonade. Each vendor must occupy one of a small number of locations from which to operate her vending services; a vending location may accom-

modate more than one vendor. What makes a vendor's decision require strategic thinking is the fact that certain vendors' services are natural *complements* to each other, whereas other services are *substitutes* for each other. Thus, if a sandwich vendor and drink vendor decide to operate in the same location, the two vendors will benefit positively from their complementary relationship. In contrast, if two sandwich vendors decide to operate in the same location, then they will split the customers due to the fact that one vendor is a substitute for the other. It is also possible that two vendors operating in the same location will have no effect on each other's sales; we call this a *neutral* relationship. Vendors operating in different locations have no effect on each other. While these general relationships apply across categories, they differ between vendors in the same category. For example, orange juice may more severely substitute for lemonade than for beer.

In modeling the full vendor game, let $\mathcal{A}$ be the set of vendors (i.e., agents), where $N = |\mathcal{A}|$ is the number of agents. Let $\mathcal{T}$ be the set of product types, where each vendor sells one product type. Let $\mathcal{S}$ be the set of possible vending locations; thus, $\mathcal{S}$ constitutes the set of pure strategies that each vendor has. Let T be a matrix that indicates how product types interact on average. For each possible ordered pair of product types $t_i, t_j \in \mathcal{T}$, $T(i,j)$ specifies the average impact a product of type t_j has on a product of type t_i. If $T(i,j) = 0$, then the product types do not interact. If $T(i,j) > 0$, then the two product types are complementary. If $T(i,j) < 0$, then the two product types are substitutes for each other.

The matrix T describes how product types interact on average. Let matrix A describe how two specific vendors interact, in particular; $A(x,y)$ gives the impact that vendor a_y has on vendor a_x. Let vendors $a_x, a_y \in \mathcal{A}$ sell products of types $t_i, t_j \in \mathcal{T}$, respectively; the value of $A(x,y)$ is obtained by sampling once from the normal distribution $N(T(i,j), \sigma^2)$, where σ^2 is a game parameter. In addition to the agent interactions, each agent a_x has an associated bias term that specifies a base-line success level for the agent independent of the impacts of other agents. Let b_x be the bias for agent a_x.

A pure strategy profile in the game is associated with the selection of a location by each agent. Let s_x represent the pure strategy choice of agent a_x. The payoff to player $a_x \in \mathcal{A}$ playing strategy $s_x \in \mathcal{S}$ is:

$$\pi_{a_x}(s_x, s_{-x}) = b_x + \sum_{a_y \in \mathcal{A}} \begin{cases} A(x,y) & \text{if } s_x = s_y \\ 0 & \text{if } s_x \neq s_y \end{cases} \quad (1)$$

Because the interaction between each pair of vendors is different, this is a many player game with no obviously exploitable structure in its exact representation. One solution method would be to construct a normal-form representation of the game and apply a standard solution algorithm. However, the size of the normal form representation is exponential in the number of vendors, so this method quickly becomes infeasible. Our approach is based on the observation that in this game, the agents naturally fall into clusters. Vendors in the same category have a similar strategic view of the game. They tend to be affected in similar ways by other vendors, and they also affect other vendors in similar ways. We next expand on this intuition.

3 The Cluster-Based Model

Our approach to compactly representing and tractably solving asymmetric N-player games is to use a (generally) lossy compression. We group a large number of agents into a much smaller number of clusters. The fundamental assumptions made by our approximation are that 1) agents clustered into the same cluster receive similar payoffs when they take the same action;[2] 2) agents clustered into the same cluster have similar influences on other agents in the same and different clusters; and 3) within each cluster, the actions of the individual agents are combined linearly. Thus, the combinatorial effects of strategic interaction are captured at the level of the cluster, which is how we realize our computational savings. In this section we introduce the cluster-based representation and explain how the model (both the cluster structure and the payoffs) is learned from data. Before continuing, let us remark that while we allow our game to be asymmetric (in payoffs) we do presently require each agent to share the same set of pure strategies $\mathcal{S}$.

3.1 Defining the Model

Let K be the number of clusters that we wish to generate; this parameter allows us to express a trade-off between fidelity to the original game and computational efficiency. Let $\mathcal{C}$ be a clustering of the N agents into K clusters, with $C_i \in \mathcal{C}$ being the ith cluster of agents. A cluster will be termed a "player" in the reduced game induced by a clustering. We reserve the term "agent" for an agent of the original N-player game.

For each combination of a cluster and a pure strategy (i.e., each element in the cross product $\mathcal{C} \times \mathcal{S}$), we construct a linear equation with $|\mathcal{S}|^K + 1$ terms. Each one of these linear equations $\hat{\pi}_{C_i}(s_x)$ is a regressor that estimates the payoff an agent in cluster C_i would receive when it plays pure strategy s_x, given the clustering $\mathcal{C}$

[2] If necessary, agents' payoffs can be normalized, according to the data, to bring payoffs into the same scale.

and probabilistic information about the strategy profile adopted by each cluster. Note that strategy s_x may be different from the strategy adopted by the cluster with which agent a_x is associated. The fact that there is one equation per cluster per strategy captures the first assumption that all agents in a cluster receive the same payoff for the same action.

One of the terms in $\hat{\pi}_{C_i}(s_x)$ is simply a constant $\beta_{C_i(s_x)}$ to capture any "offset" effects. In defining the remaining terms, let $\vec{s} = (s_1, \ldots, s_K)$ denote a strategy adopted by each cluster. Given this, then the remaining terms are of the form $\beta^{\vec{s}}_{C_i(s_x)} \Pr(\vec{s}|\mathcal{C})$, which is the product of a constant $\beta^{\vec{s}}_{C_i(s_x)}$ and the estimated probability $\Pr(\vec{s}|\mathcal{C})$ with which cluster strategy profile $\vec{s}$ is adopted by the underlying agents, given empirically observed data about the game (for all N agents) and the clustering $\mathcal{C}$. More precisely, assuming the strategies of different agents are uncorrelated, we estimate the joint probability as

$$\Pr(\vec{s}|\mathcal{C}) = \Pi_{i=1}^{K} \Pr(s_i|C_i), \tag{2}$$

where $\Pr(s_i|C_i)$ is the estimated probability that an agent in cluster C_i plays pure strategy s_i. This probability is estimated from the proportion of agents in the cluster that play strategy s_i. Thus, the payoff to an agent in a cluster depends only on the proportion of agents playing each strategy in each cluster, and not on the actions of individual agents. This captures our second assumption above. The fact that the equations are linear captures our third assumption.

Example. To concretely illustrate our regressor equations, let us assume a game where all agents have two pure strategies, L and R, and we wish to cluster agents into two clusters, A and B. We thus generate four regressors. For example, the estimated payoff obtained by an agent in cluster A playing strategy L is denoted $\hat{\pi}_A(L)$. Following this notation, the payoffs for agents in clusters A and B, using strategies L and R, are:

$$\hat{\pi}_A(L) = \beta^{L,L}_{A(L)} \cdot \Pr(L|A) \cdot \Pr(L|B) + \beta^{L,R}_{A(L)} \cdot \Pr(L|A) \cdot \Pr(R|B) + \beta^{R,L}_{A(L)} \cdot \Pr(R|A) \cdot \Pr(L|B) + \beta^{R,R}_{A(L)} \cdot \Pr(R|A) \cdot \Pr(R|B) + \beta_{A(L)} \tag{3}$$

$$\hat{\pi}_A(R) = \beta^{L,L}_{A(R)} \cdot \Pr(L|A) \cdot \Pr(L|B) + \beta^{L,R}_{A(R)} \cdot \Pr(L|A) \cdot \Pr(R|B) + \beta^{R,L}_{A(R)} \cdot \Pr(R|A) \cdot \Pr(L|B) + \beta^{R,R}_{A(R)} \cdot \Pr(R|A) \cdot \Pr(R|B) + \beta_{A(R)} \tag{4}$$

$$\hat{\pi}_B(L) = \beta^{L,L}_{B(L)} \cdot \Pr(L|A) \cdot \Pr(L|B) + \beta^{L,R}_{B(L)} \cdot \Pr(L|A) \cdot \Pr(R|B) + \beta^{R,L}_{B(L)} \cdot \Pr(R|A) \cdot \Pr(L|B) + \beta^{R,R}_{B(L)} \cdot \Pr(R|A) \cdot \Pr(R|B) + \beta_{B(L)} \tag{5}$$

$$\hat{\pi}_B(R) = \beta^{L,L}_{B(R)} \cdot \Pr(L|A) \cdot \Pr(L|B) + \beta^{L,R}_{B(R)} \cdot \Pr(L|A) \cdot \Pr(R|B) + \beta^{R,L}_{B(R)} \cdot \Pr(R|A) \cdot \Pr(L|B) + \beta^{R,R}_{B(R)} \cdot \Pr(R|A) \cdot \Pr(R|B) + \beta_{B(R)} \tag{6}$$

Since we have two agent clusters and two pure strategies, we have four regressors, each with five terms. For example, in Equation (3) we are estimating the payoff received by an agent in cluster A when it plays strategy L. Note that the third term, $\beta^{R,L}_{A(L)} \cdot \Pr(R|A) \cdot \Pr(L|B)$, represents the contribution to our agent's payoff that occurs when agents in cluster B play L in combination with agents in cluster A playing R. Thus, even though the agent for whom we are calculating the payoff is itself playing L, we are accounting for the effect that any other agents within cluster A who are playing R (in combination with cluster B agents playing L) may have upon our agent's payoff.

3.2 Model Learning

A key aspect to our work is that we do not require *a priori* knowledge of the cluster-based model. Instead, we may learn the model from observation of agent behavior and earned payoffs. We have two distinct settings in mind:

- We have access to a data set about agent actions and agent payoffs in the underlying game.
- We have access to a simulator that we can use to generate payoffs for different strategy profiles in the underlying game.

In both cases we are learning the model offline and adopting the viewpoint of an analyst, not an agent situated within the strategic environment. Notice that in the second approach we need not have a complete representation of the game, which would in general be too large to represent. Rather, all that is required is a way to generate payoffs for different strategy profiles.

From our observations we learn our cluster-based model of the underlying N-player game. Each observation consists of an N-tuple pure strategy profile and the corresponding payoffs received by each agent. We require that every action in every cluster be seen at least once in order to learn.

Given a clustering of the N agents into K clusters, we can learn the β parameters for our regressors with linear regression. Each observation of an N-agent interaction typically provides several data instances for

the linear regression. For instance, consider our example game above. Let cluster A contain N_A agents, each playing a uniform mixed strategy over pure strategies L and R. If we have M observations, then we expect to obtain $N_A M/2$ data instances each for regressor Equations (3) and (4). In this way, each additional agent that we might place into our full game will linearly increase the number of instances for the linear regression. Thus, we expect the number of observations of the full game that we need to be inversely proportional to the number of agents, at least for the purpose of estimating parameters.

Agent Clustering. We may compute a linear regression given any clustering of N agents into K clusters. Clearly, we desire the clustering $\mathcal{C}$ that provides us with the best regressors, so that we can most accurately estimate agent payoffs. We use the sum of squared errors over all agents over all regressor equations to quantify the quality of our estimations, where the error of a regressor for an agent is the difference between the estimated payoff for the agent, given the N-tuple strategy profile, and the agent's actual payoff.

Since we assume a finite number of agents, there exist a finite number of possible clusterings into K clusters; an optimal clustering must therefore exist. Nevertheless, the large number of possible clusterings precludes exhaustive search. Therefore, we use k-means clustering to obtain the clustering $\mathcal{C}$ that we use to run our linear regressions.

A key intuition in our work is that agents clustered into the same cluster receive relatively similar payoffs when they take the same action, given a shared context of what the other agents in the system do. We use this intuition to construct the features with which k-means operates. Specifically, we place each agent in an S-dimensional space, where S is the number of pure strategies available to agents. Each dimension corresponds to a pure strategy, and the location of an agent in a particular dimension is the average payoff the agent earned over our observations when it used the corresponding pure strategy. Note that, here, we are not conditioning further on what other agents do; this is to keep both dimensionality and computational complexity low. Because k-means does not always converge onto the same clustering each time it is run, we run it several times and choose the result that gives us the lowest sum of squared errors in our regressors.

4 The Twins Game

Given that we have K clusters, an obvious reduction would be to construct a K-player normal form game. We would then solve this smaller K-player game and then assign each agent within cluster i the strategy (pure or mixed) used in Nash equilibrium by the i-th player in the K-player game. This would follow the approach of Wellman et al. [11], albeit slightly extended to an asymmetric setting. But one problem with this approach is that, while the i-th player in the K-player game has no incentive to unilaterally deviate from equilibrium, cluster i cannot be treated as a true "player" because it is not a monolithic decision maker. Each cluster of agents consists of independent decision makers, whose individual incentives might not be aligned with the cluster-level incentives of the K-player game. Thus, an individual in some cluster may, in fact, wish to deviate from the prescribed K-player strategy profile.

Rather than build a K-player game, we build a $2K$-player game, where each cluster is represented twice, hence what we call a "twins game." Each cluster C is associated with two players, $\mathbb{C}$ and $\mathbb{C}'$. (Recall that a player captures a strategic entity in the clustered game and is distinct from an agent in the underlying game.) These players have multiple interpretations, depending on from whose point of view the players are being considered. When considering the payoff of player $\mathbb{C}$, $\mathbb{C}$ is interpreted as representing a single agent in cluster C, while $\mathbb{C}'$ represents the rest of the agents in the cluster in aggregate. The view from $\mathbb{C}'$ is symmetric: the payoff of player $\mathbb{C}'$ represents that of a single agent in the cluster, while $\mathbb{C}$ represents the rest of the cluster C in aggregate. From the point of view of a player $\mathbb{D}$ associated with a different cluster $D \neq C$, $\mathbb{C}$ and $\mathbb{C}'$ together represent cluster C. When $\mathbb{C}$ and $\mathbb{C}'$ adopt different strategies $s_{\mathbb{C}}$ and $s_{\mathbb{C}'}$, then for symmetry we consider that half of the agents in the cluster play $s_{\mathbb{C}}$ and half play $s_{\mathbb{C}'}$.

Under these interpretations, we can obtain the payoffs for each of the $2K$ players in the twins game for a given strategy profile. Consider the player $\mathbb{C}$ associated with cluster C, where $\mathbb{C}$ plays strategy $s_{\mathbb{C}}$ and $\mathbb{C}'$ plays strategy $s_{\mathbb{C}'}$. We instantiate the probabilities for the other players in the linear regression model, and thus define the payoff to player $\mathbb{C}$, as follows,

$$\Pr(s_i|C) = \begin{cases} 1 & \text{if } s_i = s_{\mathbb{C}'} \\ 0 & \text{otherwise} \end{cases}$$

For a cluster $D \neq C$,

$$\Pr(s_i|D) = \begin{cases} 1 & \text{if } s_i = s_{\mathbb{D}} = s_{\mathbb{D}'} \\ 1/2 & \text{if one of } s_{\mathbb{D}} \text{ or } s_{\mathbb{D}'} = s_i \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbb{D}$ and $\mathbb{D}'$ are the players corresponding to cluster D. The payoff to $\mathbb{C}$ is then $\hat{\pi}_C(s_{\mathbb{C}})$ as specified by the regressor equations. The payoff to $\mathbb{C}'$ is symmetric.

By representing each cluster twice, we can seek Nash equilibria in which the incentives of individuals within a cluster are aligned with the cluster itself. We care to locate Nash equilibria where each player and its twin use the same strategy. We will call such equilibria *twin symmetric* Nash equilibria (TSNE). A TSNE is guaranteed to exist, because there is always a twin symmetric best response to a twin symmetric strategy profile, so we can use the same argument as in Nash's proof of the existence of Nash equilibrium [9].

Let $s_{\mathbb{C}}$ denote this strategy (pure or mixed), used by both players $\mathbb{C}$ and $\mathbb{C}'$ in a TSNE (and in turn used by each underlying agent within the cluster.) From the perspective of player $\mathbb{C}$, strategy $s_{\mathbb{C}}$ is a best response to the strategies of the other players, and particularly to $\mathbb{C}'$ who also plays $s_{\mathbb{C}}$. In the construction of the twins game, the payoff to player $\mathbb{C}$ equals the payoff to an individual agent in cluster C playing $s_{\mathbb{C}}$ when the rest of cluster C plays $s_{\mathbb{C}'}$. But in a TSNE, $s_{\mathbb{C}} = s_{\mathbb{C}'}$. Thus, if $s_{\mathbb{C}}$ is the strategy recommended to $\mathbb{C}$ in a TSNE, an individual agent in cluster C will be playing a best response to its cluster as a whole playing $s_{\mathbb{C}}$. In this way, we say that our representation is *individually-responsive*. No individual agent will have incentive to deviate from the strategy recommended to its cluster.

4.1 Example

Let us continue our example from Section 3.1. We have two clusters of agents, A and B, and so will build a four player game. Let us label the players $\mathbb{A}$, $\mathbb{A}'$, $\mathbb{B}$, and $\mathbb{B}'$, where $\mathbb{A}$ and $\mathbb{A}'$ correspond to the twin players for cluster A, and similarly for the players for cluster B. Table 1 shows how we convert our regressors to a twins game. Each row of the table corresponds to a single four-player pure-strategy profile. For brevity, we show only 1/4 of the profiles, showing only those where $\mathbb{A}$ plays L and $\mathbb{A}'$ plays R. The leftmost column specifies a four-player pure-strategy profile; for example, the third row specifies a profile where players $\mathbb{A}$, $\mathbb{A}'$, $\mathbb{B}$, and $\mathbb{B}'$ play pure strategies L, R, R, and L, respectively. The remaining columns indicate the payoffs received by each player for each pure-strategy profile.

We compute player payoffs as described in the previous section. The pure strategy that a player uses in a profile determines which regressor equation we will use. For example, in the profile $LRLL$, player $\mathbb{A}$ plays L, and so we use Equation (3); player $\mathbb{A}'$ plays R, and so for this player we use Equation (4). We use Equation (5) for players $\mathbb{B}$ and $\mathbb{B}'$, since both play L.

The payoff to $\mathbb{A}$ for the profile $LRLL$ is $\beta^{RL}_{A(L)} + \beta_{A(L)}$. This is the payoff that a cluster A agent would receive if it played L in the situation where all cluster A agents actually play R (i.e., $\Pr(R|A) = 1.0$) and all cluster B agents play L. Similarly, the payoff to $\mathbb{A}'$ is $\beta^{LL}_{A(R)} + \beta_{A(R)}$, which corresponds to the payoff a cluster A agent would receive for playing R in the situation where all cluster A agents actually play L and all cluster B agents play L. More interesting is the payoff for $\mathbb{B}$, which is $\frac{\beta^{LL}_{B(L)} + \beta^{RL}_{B(L)}}{2} + \beta_{B(L)}$. This is the payoff obtained by a cluster B agent if it would play L in the situation where all cluster B agents actually play L and where half the cluster A agents play L and half play R (i.e., $\Pr(L|A) = \Pr(R|A) = 0.5$). Thus, when a player and its twin (e.g., $\mathbb{A}$ and $\mathbb{A}'$) play different pure strategies in a profile, we interpret this as a uniform distribution over the two strategies when computing the payoff for another player (e.g., $\mathbb{B}$ or $\mathbb{B}'$) in the twins game.

5 Experimental Results

In generating our observation set for the purpose of experimentation, we build N agents to play the game for some number of interactions. We provide agents with a uniform distribution over their pure strategies in order to generate an observation set with good support in the space of possible joint strategy profiles in our twins game. From these observations, and with a given number K of clusters, we learn our cluster-based approximation; this entails clustering and linear regression. We then construct the $2K$-player twins game. For comparison, we also construct a K-player game, with one player per cluster and thus without individual responsiveness. Using Gambit [8], we find all Nash equilibria of these reduced normal-form games. For each TSNE of the $2K$-player game and each NE of the K-player game, we assign the equilibrium strategies to the agents and have them interact for 100 iterations. We then calculate mean payoffs and external regret values for each agent in this new data. Note that we do not need to solve the full N-player game.

In addition to comparing results from the $2K$- and K-player games, we compare the performance of our approach with that of two model-free learning approaches. These methods are a form of reinforcement learning whereby agents play strategies that have yielded the highest mean rewards. Our first model-free approach concerns *agent-level learning* (ALL). We examine the initial observation set to determine, for each agent, which pure strategy provided the agent with the highest mean payoff. Each agent then adopts the pure strategy that provided it the highest mean payoff for use in future interactions. We generate new interaction data with agents playing these pure strategies, and then determine mean agent payoff and regret values for the new data. Our second model-free approach provides *cluster-level learning* (CLL). Here we examine the initial observation set to determine, for each

Table 1: Conversion of regression equations to normal-form "twins game" (partial specification).

Profile	$\mathbb{A}$	$\mathbb{A}'$	$\mathbb{B}$	$\mathbb{B}'$
$LRLL$	$\beta^{RL}_{A(L)} + \beta_{A(L)}$	$\beta^{LL}_{A(R)} + \beta_{A(R)}$	$\frac{\beta^{LL}_{B(L)}+\beta^{RL}_{B(L)}}{2} + \beta_{B(L)}$	$\frac{\beta^{LL}_{B(L)}+\beta^{RL}_{B(L)}}{2} + \beta_{B(L)}$
$LRLR$	$\frac{\beta^{RL}_{A(L)}+\beta^{RR}_{A(L)}}{2} + \beta_{A(L)}$	$\frac{\beta^{LL}_{A(R)}+\beta^{LR}_{A(R)}}{2} + \beta_{A(R)}$	$\frac{\beta^{LR}_{B(L)}+\beta^{RR}_{B(L)}}{2} + \beta_{B(L)}$	$\frac{\beta^{LL}_{B(R)}+\beta^{RL}_{B(R)}}{2} + \beta_{B(R)}$
$LRRL$	$\frac{\beta^{RL}_{A(L)}+\beta^{RR}_{A(L)}}{2} + \beta_{A(L)}$	$\frac{\beta^{LL}_{A(R)}+\beta^{LR}_{A(R)}}{2} + \beta_{A(R)}$	$\frac{\beta^{LL}_{B(R)}+\beta^{RL}_{B(R)}}{2} + \beta_{B(R)}$	$\frac{\beta^{LR}_{B(L)}+\beta^{RR}_{B(L)}}{2} + \beta_{B(L)}$
$LRRR$	$\beta^{RR}_{A(L)} + \beta_{A(L)}$	$\beta^{LR}_{A(R)} + \beta_{A(R)}$	$\frac{\beta^{LR}_{B(R)}+\beta^{RR}_{B(R)}}{2} + \beta_{B(R)}$	$\frac{\beta^{LR}_{B(R)}+\beta^{RR}_{B(R)}}{2} + \beta_{B(R)}$

cluster, which pure strategy provided the agents in the cluster with the highest mean payoff. Each agent in the cluster then adopts this pure strategy. We then generate new interaction data.

5.1 Vendor Game

We initialize the type-interaction matrix T as follows. Each product type is treated as a substitute for itself, thus the diagonal of T is negative. Off the diagonal, we randomly select between interaction types ($\Pr(\text{neutral}) = 0.1, \Pr(\text{substitute}) = \Pr(\text{complement}) = 0.45$), but require T to contain at least one complementary interaction. Substitution means are drawn uniformly between $[-3.0, 0.0]$, complements from $[0.0, 3.0]$, and neutral interactions have a mean of zero.

In one experiment, we use 100 agents, two agent types, two locations (L and R), $\sigma^2 = 1.5$, and 15 observations per trial over ten trials. We cluster agents into two groups (A and B). Table 2 summarizes our experiment, showing mean agent payoffs and regrets. The model-free learning methods clearly perform the worst; the strategies (always pure) they learn give agents the worst mean payoffs and highest mean regrets. The differences between the K- and $2K$-player games are most pronounced with respect to regret levels. When we examine separately the performance of pure-strategy Nash equilibria (PSNE) and mixed-strategy NE (MSNE) of the K- and $2K$-player games, we find that agents' regret levels when they use MSNE derived from K-player games are over 2.3 times as high as when they use mixed TSNE from $2K$-player games ($p < 0.004$). The K-player games often produce PSNE, and the regret levels from these PSNE are an order of magnitude higher than the TSNE of the $2K$-player game. This shows that our twins-game approach aligns the incentives of individual agents.

In another experiment, we look at different numbers of agents (N =10, 100, or 200) combined with different numbers of observations (3, 5, 10, or 15); other parameters are as above. Table 3 gives average regrets for some of our settings. Looking over all of the data, two trends appear to emerge. First, the disparities between the approaches appears to grow with the number of agents; thus, it becomes more important to have a good strategy in our game with more agents. Second, the level of regret for a given number of agents appears to consistently diminish with our approach as the number of observations increases. Nevertheless, our method consistently finds solutions that give the lowest average regret over all settings. We also ran experiments where we varied the level of noise σ^2 in agent interactions. Though the R^2 values of our regressions suffer, the performance of our approach is surprisingly robust. The general conclusion is that the model-free methods perform poorly, and that the twins game approach produces lower agent regret values than the game-form with one player per cluster.

Table 2: Results from vendor game. 100 agents, 2 types, $\sigma^2 = 1.5$, $K = 2$, 15 observations per trial. When analysis gives multiple NE in a K- or $2K$-player game, we select the NE giving the worst result. First row: mean payoffs over all agents over all trials. Significant differences are CLL vs. ALL ($p < 0.065$ paired sign-rank test), CLL and ALL vs. K- and $2K$-player games ($p < 0.014$). Second row: mean external regret over all agents over all trials. Significant differences are CLL vs. ALL ($p < 0.11$), CLL and ALL vs. K and $2K$ ($p < 0.01$), and K vs. $2K$ ($p < 0.04$).

	CLL	ALL	K-Player	$2K$-Player
Payoff	-53.98	-40.20	-17.69	**-15.50**
Regret	87.75	64.44	17.56	**3.06**

5.2 Santa Fe Bar

The second game on which we test our approach is a slight variation of the *Santa Fe bar problem* (also known as the *El Farol* bar problem) [1]. In this game, each of the N agents must independently de-

Table 3: Average regrets from vendor game for different numbers of agents and observations. First column indicates numbers of agents and observations. Note that neither agent payoffs nor regret values are normalized by N.

	CLL	ALL	K-Pl.	$2K$-Pl.
10, 3	7.6767	7.5127	4.2450	2.9837
10, 5	8.3199	6.1706	3.3321	0.5177
10, 10	7.8606	5.9436	0.7290	0.2226
100, 3	94.2589	32.4916	15.6744	4.3206
100, 5	85.0618	69.1498	10.5996	1.3062
200, 3	181.1111	122.8304	33.5588	5.4801
200, 15	151.1831	104.2937	2.0577	0.9401

cide whether or not to visit the *El Farol* bar. Unfortunately, the bar's capacity $c \in (0, 1)$ allows only $\lfloor cN \rfloor$ agents to fit comfortably. The set of pure strategies for each agent consists of two items: either the agent visits the bar, or stays at home. There are three possible outcomes for an agent: 1) the agent decides to visit the bar, but the bar is full (denoted $v^\bullet$), 2) the agent decides to visit the bar, and the bar has room (denoted v°), 3) the agent decides to stay at home (denoted h). Each agent prefers these outcomes as follows: $v^\circ \succ v^\bullet$, $v^\circ \succ h$, and $h \succ v^\bullet$. Once all N agents make their choice to visit the bar or stay at home, we obtain an N-player pure-strategy profile. The utility for agent a_x for outcome o is denoted by $U_x(o)$.

Each agent shares the same utility function, giving us a symmetric game. Since agents all share the same preferences and affect others in the same way, we cluster all agents into a single group; one cluster yields a two-player twins game to solve. Thus, this game exactly satisfies the first two assumptions of our cluster-based representation. Note, however, that it violates the third assumption: the payoffs are not linear in the number of agents playing each strategy. Unlike in the vendor game, an agent's payoff here is only one of three possible values, conditioned on whether the agent visits the bar or stays at home, and whether the total attendance exceeds the bar's capacity or not. The degree to which capacity is exceeded, or the amount of available space, does not affect an agent's payoff.

Because the Santa Fe bar game is symmetric, we can use the method of Wellman, et al., [11] (WEL) to approximate a solution. Their approach to compressing a symmetric game is to "coarse code" the profile and outcome space. They divide the agents into some number of equally sized groups. Given the constraint that all agents within a group must play the same strategy, they perform equilibrium computations using the subset of realizable strategy profiles and outcomes of the *original N-player game*. Since we compress the Santa Fe game down to a two-player twins game, the most direct comparison with WEL is to use their approach to divide the N agents into two groups of $N/2$ agents each, which also yields a two-player game. This approach is denoted by $\mathrm{W}{\downarrow_2}$. However, the approach of WEL allows for more refined approximations that use more clusters. As a point of comparison, we also consider their approach with five clusters, denoted by $\mathrm{W}{\downarrow_5}$, which involves solving a five-player game.

Table 4: Average regrets from Santa Fe bar game with different bar capacities. For capacity $c = 0.4$, the difference in regret between our method and $\mathrm{W}{\downarrow_5}$ is not statistically significant. For capacity $c = 0.5$, the differences in regret between our method, $\mathrm{W}{\downarrow_2}$ and ${\mathrm{W}{\downarrow_5}}^*$ are not statistically significant. All other differences between our approach and the others are statistically significant ($p < 0.006$ paired sign-rank test).

c	$2K$	$\mathrm{W}{\downarrow_2}$	$\mathrm{W}{\downarrow_5}$	${\mathrm{W}{\downarrow_2}}^*$	${\mathrm{W}{\downarrow_5}}^*$
0.4	0.4820	4.0	0.3508	4.0	0
0.5	0.7264	0.7750	1.6908	0	0.6690
0.6	0.8368	1.8112	0.2068	1.8080	0

Table 4 summarizes regret values obtained in experiments that examine different bar capacities $c \in \{0.4, 0.5, 0.6\}$. For all three capacity values, we set agent payoffs to be $U(v^\circ) = 4.0$, $U(v^\bullet) = -6.0$, and $U(h) = 0.0$; we use ten agents. We run ten trials for each value of bar capacity. We use 45 observations (agent interactions) per trial for learning in our approach. For complete comparison, we show the regret of the symmetric MSNE found by WEL (shown as $\mathrm{W}{\downarrow_2}$ and $\mathrm{W}{\downarrow_5}$), and also the regret of the best NE found by WEL, whether symmetric or asymmetric (shown as ${\mathrm{W}{\downarrow_2}}^*$ and ${\mathrm{W}{\downarrow_5}}^*$). Results for ALL and CLL are not shown in this table; they are significantly worse than the other approaches.

When the bar's capacity is exactly a multiple of the fraction of agents within a cluster, WEL discovers asymmetric PSNE that are actually NE of the full game. Except for this case, we see that our method is better than $\mathrm{W}{\downarrow_2}$ when the capacity is $c = 0.4$ or $c = 0.6$, and is statistically equivalent to it when $c = 0.5$. Even when we compare to WEL allowing it to use five clusters, our method performs reasonably well. It is statistically equivalent to $\mathrm{W}{\downarrow_5}$ when $c = 0.4$. When $c = 0.5$, it is statistically equivalent to the asymmetric equilibrium found by WEL (${\mathrm{W}{\downarrow_5}}^*$), and significantly better than the symmetric equilibrium ($\mathrm{W}{\downarrow_5}$). However when $c = 0.6$, $\mathrm{W}{\downarrow_5}$ is significantly better. Note that our results are consistently good despite the fact that our approach involves the added step of learning (which WEL does not).

Figure 1 shows the actual symmetric MSNE for our Santa Fe game for bar capacities $c = 0.4, 0.5, 0.6$. The figure also shows the approximations to these mixed Nash equilibria obtained with our method as well as

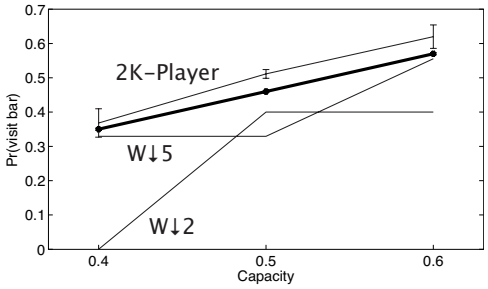


Figure 1: Symmetric mixed strategies obtained for Santa Fe bar game for bar capacities $c = 0.4, 0.5, 0.6$. X-axis indicates bar capacity, Y-axis indicates the probability with which each agent visits the bar in the mixed strategy. Data labeled "2K-Player" indicate the average mixed Nash equilibria (and standard deviation) of our 2K-player twins game over all 10 trials. Data labeled "$W\downarrow_2$" and "$W\downarrow_5$" indicate the mixed Nash equilibria obtained with WEL using two and five clusters, respectively. Bold line indicates the actual mixed Nash equilibria of the Santa Fe game.

with $W\downarrow_2$ and $W\downarrow_5$. We see that our approach is more consistently close to the actual symmetric MSNE of the Santa Fe game. Thus, our approach appears better able to detect changes in bar capacity, while at the same time requiring that only a two-player twins game be solved ($W\downarrow_2$ and $W\downarrow_5$ require two- and five-player games to be solved). Indeed, it can be show analytically that the finest distinction $W\downarrow_2$ can make is whether bar capacity is less than $N/2$ or not.

6 Conclusions

We present a method for approximating the structure of asymmetric N-player games for large N. This approximation uses a clustering approach to compress the original N-player game into a vastly smaller and more tractably solved $2K$-player game, where K is the number of groups into which we cluster the N agents. Each of the K groups of agents is associated with two players in the $2K$-player game; we call these player pairs "twins." Nash equilibria in which twins use the same strategy are "twin-symmetric" NE. We treat these NE as K-player strategy profiles that we assign to our agent groups; all agents within a group play the same strategy. The K-player NE derived from twin-symmetric NE have the property that the incentives of all agents within a cluster are aligned with the strategy assigned to that cluster. This prevents unilateral deviation by individual agents from the K-player NE. Importantly, our method does not assume knowledge of the full N-player payoff function; instead, we can learn an approximation to the payoff function through a small number of observations of agent interaction.

We test our method on two different types of game: a vendor game and the Santa Fe bar problem. We show that when agents play solutions obtained from our method, they achieve higher average payoffs and lower external regret compared with two model-free learning approaches we examine. We also compare to compression results obtained with a modified version of our approach that omits the $2K$-player game. We show that this variation is significantly less effective in providing low regret. For the Santa Fe bar problem, data show that our method provides low regret values more consistently than the compression method by Wellman, et al. [11] when using the same number of agent clusters. We also note that our approach is orthogonal to that of Wellman, et al. [11]; for example, we can divide the clusters of the twins game into subclusters to allow for asymmetric equilibria, even in symmetric games and even if learned with just one cluster.

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments. The research reported in this paper was supported in part by AFOSR grant FA9550-05-1-0321 and NSF grant DMS 0631636.

References

[1] W. B. Arthur. Inductive reasoning and bounded rationality. *Amer. Econ. Review*, 84(2):406–411, 1994.

[2] N. Bhat and K. Leyton-Brown. Computing Nash equilibria of action-graph games. In *UAI*, 2004.

[3] C. Daskalakis, A. Mehta, and C. H. Papadimitriou. Progress in approximate nash equilibria. In *8th ACM Conf. on Electronic Commerce*, 2007.

[4] C. Daskalakis and C. H. Papadimitriou. Computing equilibria in anonymous games. In *FOCS*, 2007.

[5] P. Jehiel. Analogy-based expectation equilibrium. *Journal of Economic Theory*, 123:81–104, 2005.

[6] M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. In *UAI*, 2001.

[7] M. Kearns and Y. Mansour. Efficient nash computation in large population games with bounded influence. In *UAI*, 2002.

[8] R. D. McKelvey, A. M. McLennan, and T. L. Turocy. Gambit: Software tools for game theory, version 0.2007.01.30. http://gambit.sourceforge.net, 2007.

[9] J. F. Nash. Noncooperative games. *Annals of Mathematics*, 54:189–295, 1951.

[10] Y. Vorobeychik, M. P. Wellman, and S. Singh. Learning payoff functions in infinite games. *Machine Learning*, 67:145–168, 2007.

[11] M. P. Wellman, D. M. Reeves, K. M. Lochner, S.-F. Cheng, and R. Suri. Approximate strategic reasoning through hierarchical reduction of large symmetric games. In *AAAI*, 2005.

Constrained Approximate Maximum Entropy Learning of Markov Random Fields

Varun Ganapathi
Computer Science Dept.
Stanford University
varung@cs.stanford.edu

David Vickrey
Computer Science Dept.
Stanford University
dvickrey@cs.stanford.edu

John Duchi
Computer Science Dept.
Stanford University
jduchi@cs.stanford.edu

Daphne Koller
Computer Science Dept.
Stanford University
koller@cs.stanford.edu

Abstract

Parameter estimation in Markov random fields (MRFs) is a difficult task, in which inference over the network is run in the inner loop of a gradient descent procedure. Replacing exact inference with approximate methods such as loopy belief propagation (LBP) can suffer from poor convergence. In this paper, we provide a different approach for combining MRF learning and Bethe approximation. We consider the dual of maximum likelihood Markov network learning — maximizing entropy with moment matching constraints — and then approximate both the objective and the constraints in the resulting optimization problem. Unlike previous work along these lines (Teh & Welling, 2003), our formulation allows parameter sharing between features in a general log-linear model, parameter regularization and conditional training. We show that piecewise training (Sutton & McCallum, 2005) is a very restricted special case of this formulation. We study two optimization strategies: one based on a single convex approximation and one that uses repeated convex approximations. We show results on several real-world networks that demonstrate that these algorithms can significantly outperform learning with loopy and piecewise. Our results also provide a framework for analyzing the trade-offs of different relaxations of the entropy objective and of the constraints.

1 Introduction

Markov random fields (MRFs) have become a standard tool in many applications. It is often desirable to estimate the parameters of these models from data because tuning them manually is often difficult, and learned models often exhibit better performance. While several different objectives have been proposed for the learning task, the most commonly used are maximum likelihood and maximum conditional likelihood, often with additional parameter priors (regularization penalties). These objectives cannot be optimized in closed form, but they are convex, and so the global optimum can be found using iterative methods, such as simple gradient descent or more sophisticated optimization algorithms (Minka, 2001; Vishwanathan et al., 2006). Unfortunately, each step of these optimization algorithms requires that we compute the log partition function and the gradient which in turn requires performing inference on the model with the current parameters. As MRF inference is computationally expensive or even intractable, the learning task — which executes inference repeatedly — is often viewed as intractable.

One commonly-used approach (Shental et al., 2003; Taskar et al., 2002; Sutton & McCallum, 2005) is to approximate the gradient of the maximum likelihood objective through an approximate inference technique, most often the loopy belief propagation (LBP) (Pearl, 1988; Yedidia et al., 2005) algorithm. LBP uses message passing to find fixed points of the non-convex Bethe approximation to the energy functional (Yedidia et al., 2005). Unfortunately, for some choices of models, LBP can be highly non-robust, providing wrong answers or not converging at all. These phenomena are particularly problematic when LBP is used as the inner loop of a learning algorithm, as they can lead to non-robust estimates of the gradient, and poor convergence of the entire algorithm. Other methods, such as the double loop algorithm of Yuille (2002), provide a convergent alternative to optimizing the Bethe objective. However, in the context of learning, this method would give rise to a triple-loop algorithm (with parameter updates being the outer loop), with the resulting increase in complexity. Moreover, as the inner objective is non-convex, it has multiple local minima, possibly creating problems for the more sophisticated optimizers that often assume convexity.

In this paper, we present an alternative approach, based on a unified objective that integrates both inference and learning. Our formulation starts with the dual of maximum likelihood, namely maximum entropy with expectation constraints. We approximate the entropy using the Bethe approximation and the marginal polytope with the

standard local consistency constraints. This formulation pulls all of the non-convexity into the external, unified objective, avoiding the difficulty of solving a non-convex problem in the inner loop. The overall objective, which we call *CAMEL* (Constrained Approximate Maximum Entropy Learning) is a sum of concave and convex functions. We maximize it using the iterated CCCP approach of Yuille (2002): in each iteration, we linearize the non-concave part, resulting in a constrained maximization of a concave function problem. We show how the dual of this problem can be reformulated as a sum of local logistic regressions, one for each factor in the MRF, with shared parameters. The optimization can thus be performed efficiently using state-of-the-art solvers such as conjugate gradient or L-BFGS (Zhu et al., 1997).

The unified objective provides an elegant framework to understand different approaches in terms of the approximations they make to both the objective and the constraints. In particular, we show that piecewise training method, recently proposed by Sutton and McCallum (2005), is solving a simplification of the CAMEL objective, when the linearization is chosen to be zero, and the marginal consistency constraints have been dropped. This provides a novel insight into the relationship between LBP and piecewise training, providing a direct comparison between the objectives that each is optimizing. We also show that the Uniform Propagation and Scaling algorithm of Teh and Welling (2001) is performing a coordinate-ascent procedure on the unified objective for a restricted class of models that involve no weight sharing, conditional training or regularization.

On several data sets, we find that our CCCP-based procedure has to perform relatively few relinearizations. On tasks where LBP works well in the inner loop, we show that our method achieves comparable results. On a vision task, where LBP fails completely, our method still performs well, and outperforms piecewise training by a large margin.

2 Background and Related Work

2.1 Maximum Entropy Learning

For the purposes of this paper, we focus on discrete MRFs, and utilize the log-linear representation, which most naturally accommodates rich feature spaces, regularization, and parameter sharing. A log-linear MRF encodes a joint distribution over assignments $\boldsymbol{x}$ to a set of (discrete) random variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$. We assume that the distribution is defined in terms of a set of *features* f_l, each with its own *weight* w_l. In our formulation, features can be shared both within and between factors in the network. We define $\boldsymbol{f}_l$ to be the overall sufficient statistic associated with f_l. For standard (generative) MRFs, the distribution defined by the log-linear model is:

$$\mathrm{P}_{\boldsymbol{w}}(\boldsymbol{x}) = \frac{1}{Z}\prod_l \exp\left(w_l \boldsymbol{f}_l(\boldsymbol{x})\right) = \frac{1}{Z}\exp(\boldsymbol{w}^T \boldsymbol{f}(\boldsymbol{x})), \tag{1}$$

where Z is the normalizing partition function.

Given a training set $\mathcal{D}$ with instances $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^M$, we can choose the parameters $\boldsymbol{w}$ to maximize the log-likelihood $\log P(\mathcal{D} \mid \boldsymbol{w}) = \sum_{m=1}^{M} \log P(\boldsymbol{x}^m \mid \boldsymbol{w})$. This objective has no closed form solution, but it is convex, and so can be optimized using simple gradient descent, or more sophisticated optimization techniques such as conjugate gradient or L-BFGS (Zhu et al., 1997).

It is well-known that the dual of maximum likelihood is maximum entropy (Berger et al., 1996), subject to moment matching constraints on the expectations of features taken with respect to the distribution.

$$\begin{array}{rl} \text{maximize}_Q & \mathrm{H}_Q(\boldsymbol{X}) \\ \text{subject to} & \mathrm{E}_Q[\boldsymbol{f}] = \mathrm{E}_{\hat{\mathrm{P}}}[\boldsymbol{f}] \\ & \sum_{\boldsymbol{x}} Q(\boldsymbol{x}) = 1 \\ & Q(\boldsymbol{x}) \geq 0 \end{array} \tag{2}$$

This derivation also generalizes to the case of optimizing a conditional-likelihood objective, as in conditional random fields (CRFs) (Lafferty et al., 2001). Here, we encode a conditional probability distribution over a set of variables $\boldsymbol{Y}$ given an observed set of variables $\boldsymbol{X}$: $\mathrm{P}_{\boldsymbol{w}}(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})}\prod_l \exp(w_l f_l(\boldsymbol{y}_l, \boldsymbol{x}_l))$, where $Z(\boldsymbol{x})$ is a normalizing constant that defines a distribution over $\boldsymbol{y}$ for every value of $\boldsymbol{x}$. The learning task for a CRF is to maximize the conditional likelihood of the training data D, $\max_{\boldsymbol{w}} \sum_{(\boldsymbol{x}^m, \boldsymbol{y}^m) \in D} \log P_w(\boldsymbol{y}^m \mid \boldsymbol{x}^m)$. The dual of this problem is the sum of the entropies of the conditional distributions, subject to expectation constraints (McCallum et al., 2000):

$$\begin{array}{rl} \text{maximize}_Q & \sum_m \mathrm{H}_Q(\boldsymbol{Y} \mid \boldsymbol{x}^m) \\ \text{subject to} & \sum_m \mathrm{E}_{Q_{\boldsymbol{Y}\mid\boldsymbol{x}^m}}[\boldsymbol{f}(\boldsymbol{Y}, \boldsymbol{x}^m)] = \sum_m \boldsymbol{f}(\boldsymbol{y}^m, \boldsymbol{x}^m) \\ & \sum_{\boldsymbol{y}} Q_{\boldsymbol{Y}\mid\boldsymbol{x}^m}(\boldsymbol{y}) = 1, \forall m \\ & Q(\boldsymbol{y} \mid \boldsymbol{x}^m) \geq 0,\ \forall m, \boldsymbol{y} \end{array} \tag{3}$$

We can also generalize this formulation to encompass regularization of parameters (or, equivalently, a parameter prior). For example, regularizing the L_2 norm of the weights $\boldsymbol{w}$ corresponds to replacing the hard constraints equating the expected and empirical counts with an L_2 penalty on their difference in the objective:

$$\begin{array}{rl} \text{maximize}_Q & \sum_m \mathrm{H}_Q(\boldsymbol{Y} \mid \boldsymbol{x}^m) - \lambda \|\mathrm{E}_Q[\boldsymbol{f}] - \mathrm{E}_{\hat{\mathrm{P}}}[\boldsymbol{f}]\|_2 \\ \text{subject to} & \sum_{\boldsymbol{y}} Q_{\boldsymbol{Y}\mid\boldsymbol{x}^m}(\boldsymbol{y}) = 1, \forall m \\ & Q(\boldsymbol{y} \mid \boldsymbol{x}^m) \geq 0,\ \forall m, \boldsymbol{y} \end{array} \tag{4}$$

For simplicity of exposition and without loss of generality, we present our derivation for the case of generative training without regularization.

2.2 Constrained Entropy Approximation

Unfortunately, the entropy-based optimization problems presented above are no more tractable than their maximum likelihood dual, as they require that we optimize over the space of distributions Q, whose dimension is exponentially large. A tractable approximation to this problem can be obtained by applying the same sequence of transformations used by Yedidia et al. (2005) to derive the LBP message-passing inference algorithm as a fixed point to the problem of optimizing objective. More precisely, we consider a *cluster graph* consisting of a set of clusters $\{C_i\}$, where each has a scope $\boldsymbol{c}_i \subset \{X_1, \ldots, X_n\}$. Pairs of clusters are connected by edges (C_i, C_j), annotated with sepsets $\boldsymbol{s}_{ij} \subseteq \boldsymbol{c}_i \cap \boldsymbol{c}_j$.

Now, rather than optimizing over the space of distributions Q, we optimize over the set of *pseudo-marginals* — a set $\boldsymbol{\pi} = \{\pi_i(\boldsymbol{c}_i) \,:\, C_i \in \mathcal{C}\}$, $\boldsymbol{\mu} = \{\mu_{ij}(\boldsymbol{s}_{ij}) \,:\, (C_i, C_j) \in \mathcal{C}\}$, subject to local (cluster-based) calibration and normalization constraints:

$$\begin{array}{rcll} \mu_{ij}(\boldsymbol{s}_{ij}) & = & \sum_{\boldsymbol{c}_i \setminus \boldsymbol{s}_{ij}} \pi_i(\boldsymbol{c}_i) & \forall (C_i, C_j) \in \mathcal{C} \\ \sum_{\boldsymbol{c}_i} \pi_i(\boldsymbol{c}_i) & = & 1 & \forall C_i \in \mathcal{C}. \end{array} \tag{5}$$

The local marginal consistency constraints are a relaxation (outer bound) on the *marginal polytope* (Wainwright et al., 2003) — the set of all $\boldsymbol{\pi}, \boldsymbol{\mu}$ that can be obtained by marginalizing out a legal distribution $Q(X_1, \ldots, X_n)$. That is, whereas all legal marginals satisfy the constraints of Eq. (5), there are assignments $\boldsymbol{\pi}, \boldsymbol{\mu}$ that satisfy these constraints but are not the marginals of any legal distribution.

Continuing as in the LBP derivation, we approximate the entropy $\mathrm{H}_Q(\boldsymbol{X})$ as:

$$\mathrm{H}_Q(\boldsymbol{X}) \approx \sum_{C_i \in \mathcal{C}} \mathrm{H}_{\pi_i}(C_i) - \sum_{S_{ij} \in \mathcal{C}} \mathrm{H}_{\mu_{ij}}(S_{ij}) \tag{6}$$

This reformulation is exact when the cluster graph is a tree, but is approximate otherwise. We note that one can also use other approximations to the entropy that are concave, e.g., those proposed by Wainwright et al. (2003), Weiss et al. (2007). We focus our discussion mostly on the Bethe approximation, which often provides a better approximation, if it converges.

Putting these approximations together, we obtain the following *CAMEL* objective (Constrained Approximate Maximum-Entropy Learning):

$$\begin{array}{rl} \text{maximize}_{\pi} & \sum_i \mathrm{H}(\pi_i) - \sum_{ij} H(\sum_{\boldsymbol{c}_i \setminus \boldsymbol{s}_{ij}} \pi_i(\boldsymbol{c}_i)) \\ \text{subject to} & (\mathrm{a}) \mathrm{E}_{\boldsymbol{\pi}}[\boldsymbol{f}] = \mathrm{E}_{\hat{\mathrm{P}}}[\boldsymbol{f}] \\ & (\mathrm{b}) \sum_{\boldsymbol{c}_i \setminus \boldsymbol{s}_{ij}} \pi_i(\boldsymbol{c}_i) = \sum_{\boldsymbol{c}_j \setminus \boldsymbol{s}_{ij}} \pi_j(\boldsymbol{c}_j) \\ & (\mathrm{c}) \sum_{\boldsymbol{c}_i} \pi_i(\boldsymbol{c}_i) = 1 \\ & \pi \geq 0 \end{array} \tag{7}$$

To illustrate this equation, consider a simple MRF over the binary variables A, B, C, with three clusters AB, BC, AC. We assume that the log-linear model is defined by the following two features, both of which are shared over all clusters: $f_{00}(x, y) = 1$ if $x = 0$ and $y = 0$, and 0 otherwise; and $f_{11}(x, y) = 1$ if $x = 1$ and $y = 1$. Assume we have 3 data instances $[0, 0, 1]$, $[0, 1, 0]$, $[1, 0, 0]$. The unnormalized empirical counts of each feature, pooled over all clusters, is then $E_{\hat{P}}[f_{00}] = (1 + 1 + 1)/3 = 1$, $E_{\hat{P}}[f_{11}] = 0$. In this case Eq. (7) would take the following form:

$$\begin{array}{rl} \text{maximize}_{\pi} & \mathrm{H}(\pi_{AB}) + \mathrm{H}(\pi_{BC}) + \mathrm{H}(\pi_{AC}) \\ & -\mathrm{H}(\pi_A) - \mathrm{H}(\pi_B) - \mathrm{H}(\pi_C) \\ \text{subject to} & \sum_i E_{\pi_i}[f_{00}] = 1 \\ & \sum_i E_{\pi_i}[f_{11}] = 0 \\ & \sum_a [\pi_{AB}] - \sum_c [\pi_{BC}] = 0 \\ & \sum_b [\pi_{BC}] - \sum_a [\pi_{AC}] = 0 \\ & \sum_c [\pi_{AC}] - \sum_b [\pi_{AB}] = 0 \\ & \sum_{\boldsymbol{c}_i} \pi_i(\boldsymbol{c}_i) = 1 \quad i = 1, 2, 3 \\ & \pi \geq 0 \end{array} \tag{8}$$

2.3 Related Work

A formulation similar to the CAMEL objective of Eq. (7) was used by Teh and Welling (2001) to derive Uniform Propagation and Scaling(Teh & Welling, 2001). They begin with minimizing the Kullback-Liebler divergence and then present an approximation motivated by the LBP derivation. They then use a Lagrange-multiplier analysis to define the fixed points for their objective, and obtain a unified message passing algorithm. Their formulation, however, is highly restricted in the following ways: (1) it only covers parametrizations that are full factors over the clusters in the network, without shared parameters; (2) it only allows moment matching constraints over features involving single variables; (3) it does not cover conditional training; (4) it does not allow for regularization. These assumptions are violated in virtually any practical MRF application. Moreover, their unified message passing algorithm is based on iterated proportional fitting, and thereby performs coordinate ascent on the Largrange multipliers. This technique works reasonably well when the constraints tie together few variables, as in their restricted class of problems. In the general case with shared parameters, coordinate ascent will perform poorly (Minka, 2001) because iterating one constraint will often cause a large number of other constraints to be violated.

The unified CAMEL objective was also proposed by Wainwright (2006). However, Wainwright's optimization approach differs from ours in that he selects a single convex approximation to the entropy, and then uses a standard double-loop algorithm which optimizes log-likelihood in the outer loop using a convexified LBP as the inference engine in the inner loop. By contrast, our approach optimizes the original Bethe approximation, using the iterated CCCP approach. An analysis of the unified objective is also used by Wainwright et al. (2003) to show that maximum likelihood learning with LBP has a simple closed form solution (called pseudo-moment matching), in the very re-

stricted setting of generative MRFs, with full table factors, no regularization, and no shared parameters. (This solution is obtained by estimating the cluster beliefs directly as the marginals of the empirical distribution, and then using a simple procedure for computing the parameterization from those marginals.)

Also interesting is the connection between Eq. (7) and the piecewise training approach of Sutton and McCallum (2005). If we further relax the optimization problem by removing the marginal consistency constraints in (b), and further approximate the objective by eliminating the negative entropy terms on the right-hand side, we obtain the following optimization problem:

$$\begin{array}{ll} \text{maximize}_{\pi} & \mathrm{H}_{\pi_i}(C_i) \\ \text{subject to} & \mathrm{E}_{\boldsymbol{\pi}}[\boldsymbol{f}] = \mathrm{E}_{\hat{\mathrm{P}}}[\boldsymbol{f}] \end{array} \tag{9}$$

This optimization problem is concave, and its dual is precisely the piecewise objective. This equivalence highlights the two important differences between the objective of piecewise training and that of maximum likelihood using LBP inference: piecewise training uses a particular form of concave entropy approximation, and omits any attempt to enforce the marginal consistency constraints between clusters. We study the implications of these approximations.

3 Optimization

We now consider the task of optimizing the Eq. (7) objective. The standard strategy is to introduce Lagrange multipliers, or weights, for the expectation constraints. For a fixed set of weights, the resulting optimization problem is minimization of the Bethe Free Energy where the potentials are defined by the setting of the weights. The standard optimization strategy is a *double loop* method, where the outer loop takes gradient steps relative to the weights, and the inner loop computes the gradient using a message passing algorithm such as LBP. As we shall demonstrate in our experiments, this method can run into significant difficulties in complex models, as having a non-convex problem as the inner loop of outer optimization creates convergence problems.

The objective in Eq. (7) has two components: the positive clique entropies $\sum_i \mathrm{H}(\pi_i)$, and the negative sepset entropies $-H(\sum_{\boldsymbol{c}_i \setminus \boldsymbol{s}_{ij}} \pi_i(\boldsymbol{c}_i))$. The positive entropies are concave functions, whereas the negative entropies are convex. For the Bethe approximation, this objective is not concave, and therefore must be optimized with care. The concave entropy approximations mentioned above can be optimized directly using standard optimization methods.

Our strategy for the non-concave case is based on the CCCP approach of Yuille (2002), which was developed for the approximate inference task. We write the objective as a sum of a concave function and a convex function. We then linearize the convex component $-\sum_{ij} \mathrm{H}(\sum_{\boldsymbol{c}_i \setminus \boldsymbol{s}_{ij}} \pi_i(\boldsymbol{c}_i))$ with a linear function $g^T \pi$. This approximation gives rise to a problem where we maximize a concave function subject to constraints. This problem can be solved efficiently using standard optimization methods. We now relinearize the convex component around the solution found, and repeat the process.

We note that there are several other ways to "carve out" a concave portion of this objective. In particular, more elaborate methods can be defined based on the concave entropy approximations of Wainwright et al. (2003), Heskes (2006), Weiss et al. (2007). These entropy approximations use *counting numbers* to define a weighted sum of positive and negative entropies that is guaranteed to be concave overall. Thus, we might be able to "fold in" a portion of the negative entropy terms into the positive entropy terms, while still leaving the sum concave. Here, for simplicity, we consider the simplest approach, where we take only the positive entropies to be concave, and (iteratively) linearize the negative entropies. The analysis of Yuille shows that this process is guaranteed to converge to a local optimum of the original objective.

In the remainder of this section, we first discuss the inner loop of this algorithm — the solution of the constrained convex optimization problem and then present the iterated linearization procedure. In each iteration, our optimization problem now takes the following form:

$$\begin{array}{ll} \text{maximize}_{\pi} & \sum_i \mathrm{H}(\pi_i) - \sum_i g_i^T \pi_i \\ \text{subject to} & (\mathrm{a}) \mathrm{E}_{\boldsymbol{\pi}}[\boldsymbol{f}] = \mathrm{E}_{\hat{\mathrm{P}}}[\boldsymbol{f}] \\ & (\mathrm{b}) \sum_{\boldsymbol{c}_i \setminus \boldsymbol{s}_{ij}} \pi_i(\boldsymbol{c}_i) = \sum_{\boldsymbol{c}_j \setminus \boldsymbol{s}_{ij}} \pi_j(\boldsymbol{c}_j) \\ & (\mathrm{c}) \sum_{\boldsymbol{c}_i} \pi_i(\boldsymbol{c}_i) = 1 \\ & \pi \geq 0 \end{array} \tag{10}$$

In this equation, the objective is the sum of entropies of the cliques, with a linear contribution that does not affect the form of the optimization.

We want to solve this optimization problem via its dual; if we ignore the marginal consistency constraints, the dual would simply be a product of log-linear estimation problems with shared parameters. Appealingly, we can also accommodate the marginal consistency constraints within this framework by introducing auxiliary features that capture violations of marginal consistency. In particular, we arbitrarily select a directionality (i, j) for each edge ij in the cluster graph, and for each one define a set of features $h_{ij}^{\boldsymbol{s}_{ij}}$, one for each assignment $\boldsymbol{s}_{ij}$ to the sepset; this feature measures violation of marginal consistency for this particular sepset assignment. The scope of these features is C_i and C_j. Applied to an assignment $\boldsymbol{c}_i$, it takes the value $+1$ if $\boldsymbol{c}_i$ agrees with $\boldsymbol{s}_{ij}$; applied to $\boldsymbol{c}_j$, it takes the value -1 if if $\boldsymbol{c}_j$ agrees with $\boldsymbol{s}_{ij}$. Thus, the overall value of the feature is 0 if the cliques agree on the assignment $\boldsymbol{s}_{ij}$, and ± 1 otherwise. The expected value of these features is

$$\mathrm{E}_{\boldsymbol{\pi}}[h_{ij}^{\boldsymbol{s}_{ij}}] = \sum_{\boldsymbol{c}_i \setminus \boldsymbol{s}_{ij}} \pi_i(\boldsymbol{c}_i) - \sum_{\boldsymbol{c}_j \setminus \boldsymbol{s}_{ij}} \pi_j(\boldsymbol{c}_j).$$

The expected value of these features on the empirical distribution, $\mathrm{E}_{\hat{P}}[h_{ij}]$, is necessarily 0, as all valid distributions have consistent marginals.

Continuing our running example, here we would have one set of features for the B sepset that relates the AB and BC clusters. Let us consider the feature that measures the disagreement on the $B = 0$. The feature $h_{12}^{B=0}$ is defined as follows: when applied to AB, it returns 1 whenever $B = 0$; when applied to BC, it returns -1 whenever $B = 0$; in all other cases, its value is 0. Then, the expectation of this feature on the factors will precisely be $\pi_{AB}(00) + \pi_{AB}(10) - \pi_{BC}(00) - \pi_{BC}(01)$. As each π_i is constrained to be normalized, the expected value of $h_{12}^{B=0}$ is the degree to which π_{AB} and π_{BC} disagree on $P(B = 0)$.

We can now rewrite Eq. (7) as standard maximum entropy subject to expectation constraints:

$$\begin{array}{rl} \text{maximize}_{\pi} & \sum_i \mathrm{H}(\pi_i) - \sum_i g_i^T \pi_i \\ \text{subject to} & (\mathrm{a})\mathrm{E}_{\boldsymbol{\pi}}[\boldsymbol{f}] = \mathrm{E}_{\hat{\mathrm{P}}}[\boldsymbol{f}] \\ & (\mathrm{b})\mathrm{E}_{\boldsymbol{\pi}}[\boldsymbol{h}] = 0 \\ & (\mathrm{c})\sum_{\boldsymbol{c}_i} \pi_i(\boldsymbol{c}_i) = 1 \\ & \pi \succeq 0 \end{array} \tag{11}$$

We compute the Lagrange dual of this problem by introducing two sets of Lagrange multipliers: $\boldsymbol{w}$ for the original expectation constraints, and $\boldsymbol{\delta}$ that correspond to the new marginal consistency features h. For a cluster assignment $\boldsymbol{c}_i$ to C_i and a feature f_l, we define $f_l(\boldsymbol{c}_i)$ to be the portion of the sufficient statistics of f_l derived from the C_i portion of the overall joint assignment to the network variables. We can now define:

$$\pi_i(\boldsymbol{c}_i) = \frac{1}{Z_i} \exp(\boldsymbol{w}^T \boldsymbol{f}(\boldsymbol{c}_i) + \sum_{(i,j)\in\mathcal{C}} \sum_{\boldsymbol{s}_{ij}} \delta_{ij}^{\boldsymbol{s}_{ij}} h_{ij}^{\boldsymbol{s}_{ij}}(\boldsymbol{c}_i) + g_{\boldsymbol{c}_i})$$

where Z_i is the normalization constant. The dual of Eq. (11) can now be written as:

$$\underset{\boldsymbol{w},\boldsymbol{\delta}}{\text{maximize}} \quad \sum_{C_i\in\mathcal{C}} \left[\sum_{m=1}^{M} \sum_{l} w_l f_l(\boldsymbol{c}_i^m) - \ln Z_i \right] \tag{12}$$

Continuing our 3-cluster example, assuming $g = 0$, the dual objective is written as the sum of three terms, one for each cluster. The term for the AB cluster with our three data points, would be

$$w_{00} \sum_{m=1}^{3} f_{00}(a^m, b^m) + w_{11} \sum_{m=1}^{3} f_{11}(a^m, b^m) - 3 \log Z_1$$

Our overall objective would be the sum of the three cluster terms, over the parameters w_{00}, w_{11} and $\delta_{12}^0, \delta_{12}^1, \delta_{13}^0, \delta_{13}^1, \delta_{23}^0, \delta_{23}^1$. We can then define the potentials using the derived parameters; for example, $\pi_1(A, B) = \exp(w_{00} f_{00}(A, B) + w_{11} f_{11}(A, B) + \sum_b \delta_{12}^b h_{12}^b(AB) + \sum_a \delta_{13}^a h_{13}^a(AB))$.

We see that Eq. (12) is the sum of local likelihoods, one per training instance $\boldsymbol{x}^m$ and cluster C_i in the cluster graph. In fact, this objective is very similar to both the piecewise objective and standard multiclass logistic regression. The linear term g produces a bias term for each cluster. As in piecewise training, we obtain a set of log-linear estimation problems that are coupled via the use of shared parameters. Unlike piecewise training, our formulation enforces marginal consistency constraints through the use of the Lagrange multiplier $\boldsymbol{\delta}$. We are guaranteed that the factors resulting from this optimization will be a set of *consistent* pseudo-marginals. Thus, our learning objective couples inference — obtaining a set of consistent marginals — with learning.

After fully optimizing the concave subproblem, we update the linearization as per the CCCP algorithm:

$$\begin{aligned} g_{\boldsymbol{c}_i} &:= \frac{\partial}{\partial \pi_i(\boldsymbol{c}_i)} \sum_{ij} \mathrm{H}\Big(\sum_{\boldsymbol{c}_i \setminus \boldsymbol{s}_{ij}} \pi_i(\boldsymbol{c}_i) \Big) \\ &= \sum_{i \to j} \left(-1 - \log\Big(\sum_{\boldsymbol{c}_i \setminus \boldsymbol{s}_{ij}} \pi_i(\boldsymbol{c}_i) \Big) \right) \end{aligned}$$

We repeat the procedure of optimizing the concave subproblem and relinearizing until the change in linearization term g is small. This procedure is guaranteed to converge to a fixed point of the original objective (Yuille, 2002). We experimented with initializing g by evaluating the above expression at $\pi = \hat{\pi}$.

We note that, since at convergence the clusters agree on their separators, we could also use another linear combination of the marginals of the clusters to define the negative entropy. One reasonable choice is to define the sepset marginal for each edge as half the marginal of the source cluster and half of the target cluster, thus splitting the linearization evenly between the two clusters.

4 Experimental Results

The CAMEL objective consists of an objective which is a sum of a convex and concave entropy approximation and constraints which include the moment matching and local marginal consistency. We tested algorithms where we explored dropping the negative entropy terms and dropping the marginal consistency constraints in order to investigate the effects independently.

- Piecewise Training (Sutton & McCallum, 2005) which is equivalent to the CAMEL objective with the local consistency constraints and negative entropy terms removed.
- CAMEL(0), which is the CAMEL objective with the negative entropies removed, which corresponds to Piecewise Training that respects the marginal consistency constraints. We first run Piecewise, and then

introduce the marginal consistency constraints as we found this ran more quickly.

- LBP, approximating the gradient using Residual Belief Propagation (Elidan et al., 2006). We tested various convergence criteria and show the choice that worked best. When we change the weights, we restart inference from the final messages from the previous setting of which improves the speed and robustness of the learning algorithm.
- CCCP CAMEL which is the algorithm proposed in the paper, initialized at $g = 0$.
- CCCP CAMEL(Empirical), initialized from the empirical distribution as discussed earlier.

All of the objective functions are optimized using a limited-memory BFGS (Zhu et al., 1997), using the recommended parameters that come with the standard software package. At test time, we use RBP for inference.

We test the performance of each algorithm by training conditional random fields on the following three data sets: the CONLL 2003 English named entity recognition data set, the CMU Seminar Announcements information extraction data set, and the COREL image segmentation data set. We use the Gaussian prior of $\sigma^2 = 10$ that was used in previous work for the NLP data sets, and no regularization for the vision data set, which has a very small number of parameters relative to the number of training instances. For both NLP tasks, our features were calculated as in Finkel et al. (2005) and we evaluated our results by calculating the macro F1 and micro F1 scores.

The CONLL data set[1] is a collection of Reuters newswire articles annotated with four entity types: person, location, organization, and miscellaneous. The task is to correctly classify tokens as the correct entity type. We trained on the standard training set of 947 documents and tested on the standard test set of 231 documents. The CRF model for this task is the skip chain model developed by Sutton and McCallum (2004), which is the standard linear-chain model for information extraction, augmented with long-range dependencies between identical capitalized tokens.

Our second task is to extract information about seminars from the email announcements data set of Freitag (1998) [2]. The data is labeled with four fields: Start-Time, End-Time, Location and Speaker. It consists of 485 emails containing seminar announcements at Carnegie Mellon University. The CRF model for this task is the skip-chain model developed by Sutton and McCallum (2004). We evaluated the performance averaged over four folds.

Our final task is an image segmentation task in the Corel data set. These images are pre-segmented into seven classes: rhino, polar bear, water, snow, vegetation, sky, and ground. Here, we used a newly developed model (Gould et al., 2008) constructed as follows: Pixels are first grouped into a number of super-pixels, using the normalized min-cut algorithm of Ren and Malik (2003). Seven (flat) boosted classifiers were trained to learn the singleton potentials for each patch in terms of its appearance features. Edge potentials were a full (symmetric) 7×7 table. A CRF was then trained to learn both the edge potentials, the weights of the pre-learned node potentials, and the bias term. Thus, in this model, there is a total of 56 (8×7) parameters for the node potentials and 21 parameters for the edge potentials. Our evaluation metric is the accuracy of the classifier at predicting the class of each super-pixel. We randomly partitioned the 80 images into three folds of 40 training and 40 test images and averaged the results.

On the named entity recognition task, all of the methods perform reasonably well. The LBP method and the CCCP CAMEL method perform the best and almost identically. The table also presents the running times, but these are very sensitive to the convergence criteria used for the inner loop, which affect the different algorithms in unpredictable way. Thus, these running times should be viewed only as a very coarse guideline to the orders of magnitude. The number of outer loops of the CCCP algorithms was small each time, illustrating that a small number of relinearizations is required to reach convergence.

On the information extraction task, loopy belief propagation performs the best, but the difference between the algorithms is not much larger than the standard deviation of the estimate. On this data set, after eliminating features that appear only once, there are 346,140 features, and 3 million weights, and therefore the additional features and weights added by the CAMEL variants (1 million) are a small proportion; as a result, the additional running times for CAMEL(0) over Piecewise is not significant. However, due to the large number of features, the learned distribution is very similar to the empirical distribution, so that the marginal consistency constraints are automatically satisfied. Indeed, the CAMEL version that initializes from the Piecewise Training results converges very quickly.

ALGORITHM	TIME(S)	ACCURACY(%)
PIECEWISE TRAINING	911(20)	78.6(1.3)
CAMEL(0)	4082(267)	82.9(0.9)
LBP	4437(307)	53.2(4.2)
CCCP CAMEL	40501(3994)	**84.4(.6)**
CCCP CAMEL(EMPIR)	37032(4419)	**84.4(.5)**

Table 3: Results for the image segmentation task; results are averaged over 3 folds, with standard deviation in parens.

On the image segmentation task, the story is quite different. This data set has only 84 weights, with very extensive sharing across all the clusters. Learning with LBP fails badly, as the optimizer is incapable of making any progress in op-

[1] Available at http://cnts.uia.ac.be/conll2003/ner/

[2] Available at http://nlp.shef.ac.uk/dot.kom/resources.html

ALGORITHM	OUTER LOOPS	INNER LOOP	TIME(S)	MACRO F1	MICRO F1
PIECEWISE TRAINING	1	82	7348	83.7	85.2
CAMEL(0)	1	20	9348	83.9	85.5
LBP	3	97,20,12	13871	84.6	86.2
CCCP CAMEL	2	250,41	40000	84.6	86.1
CCCP CAMEL(EMPIRICAL)	2	150,40	24833	84.5	85.9

Table 1: Results for the named entity recognition task.

ALGORITHM	OUTER LOOPS	MACRO F1	MICRO F1
PIECEWISE TRAINING	0	88.5(1.0)	88.5(.9)
CAMEL(0)	0	88.5(1.0)	88.5(.9)
LBP	0	89.6(1.1)	89.6(1.0)
CCCP CAMEL	2	88.7(1.1)	88.6(1.0)
CCCP CAMEL(EMPIRICAL)	1	88.7(1.1)	88.7(1.0)

Table 2: Results for the information extraction task; results are averaged over 4 folds, with standard deviation in parens.

timizing the objective. We tried to restart the optimizer several times, which we found helped significantly on the NLP data sets, but this was not effective on the image segmentation data set. This is not surprising because this model has very many tight loops, which may cause LBP to be highly unstable. The CAMEL algorithms achieve the best performance on this data set. The large margin between Piecewise and CAMEL illustrates that enforcing the marginal consistency constraints is beneficial. In addition, the improvement of CCCP CAMEL over CAMEL(0) is also significant, demonstrating that the negative entropy terms are also important. The CCCP Camel algorithms had essentially converged after seven linearizations on all folds.

A different view of these results can be obtained from the optimization perspective. In the maximum entropy problem, sharing parameters corresponds to reducing the number of rows of the moment-matching constraint matrix, thereby reducing its rank. The lower the rank, the less constrained the optimization problem, and thus the more likely that the distribution will deviate from the marginal polytope. In the NLP data set, this did not occur because there are more weights than data instances, so that the moment-matching constraint matrix is close to full-rank. Thus, the additional local-consistency constraints have little effect on the rank, and therefore do not further constrain the problem significantly. By contrast, in the vision domain, the graph has many shared parameters, so that the problem is highly under-constrained, and it is easy to leave the marginal polytope while still agreeing with the empirical moments. By adding the many constraints that result from the marginal consistency constraints in this (fairly dense) graph, we provide important information that moves the solution towards a more plausible region of the space.

5 Discussion and Conclusions

The main contribution of our paper is a convergent algorithm to perform learning with the Bethe approximation that only has to solve convex problems in the innermost loop. As in the work of Teh and Welling (2001), Wainwright (2006), we use the entropy-dual of the maximum likelihood task, and approximate it in ways that parallel the derivation of the belief propagation algorithm (Yedidia et al., 2005). However, unlike the formulation of Teh and Welling, our approach encompasses parameter sharing, regularization, and conditional training, all of which are ubiquitous in real-world problems. Unlike the work of Wainwright, our algorithm optimizes the Bethe approximation to the objective, rather than a convex surrogate.

Our optimization algorithm uses an outer loop where we repeatedly linearize the unified, non-convex objective, and an inner loop where we optimize the linearized objective. Our inner loop reformulates the marginal consistency constraints using special features, allowing it to use standard gradient-based methods to find a set of *consistent* pseudo-marginals. Thus, our inner loop jointly solves a problem that unifies learning and inference (obtaining a set of consistent marginals). We note that the inner loop can also be solved using other methods; for example, we can optimize our convex subproblems by performing maximum likelihood estimation using sufficient statistics computed by a convex-BP message passing algorithm (Weiss et al., 2007). The most efficient method is likely to depend on the details of the problem (Minka, 2001), such as the extent to which parameters are shared across the network. Studying the relative trade-offs of different scheduling algorithms is a useful direction for future work.

Our unified objective provides a framework for understanding learning algorithms in terms of the approximations they make both on the objective and on the constraints. For example, we saw that it allows us to reinterpret the piecewise training method of Sutton and McCallum (2005) in terms of a particular approximate objective and a particular set of constraints, providing new insight on the connection between piecewise training and LBP. We note that previous work relating the two (Sutton & Minka, 2006) does so on a procedural basis, viewing piecewise as a variant of LBP that does not pass any messages.

This framework also allows us to evaluate the relative value

of different approximations to both the objective and the constraints. In particular, our results show that even a simple linearization of the objective (just dropping the negative entropy terms) provides results comparable to the full Bethe approximation. By contrast, we show that significant gains are obtained by incorporating the marginal consistency constraints. In retrospect, this finding is perhaps not surprising, as this constraint encodes a true bias about real-world probability distributions. It does, however, suggest that additional performance gains may be obtained by further reducing the space of allowable pseudo-marginals by incorporating additional constraints that are true for marginals of a valid probability distribution (Wainwright & Jordan, 2006; Sontag & Jaakkola, 2007). Thus, the unified view of learning as constrained optimization of an approximate entropy function enables an exploration of the space of approximations for both the objective and the constraints, a perspective that has dramatically improved the performance of approximate inference over the past few years.

Acknowledgments

This work was supported by the Office of Naval Research under MURI N000140710747 and by the Boeing Corp. We thank Ben Packer for helpful discussions.

References

Berger, A., Pietra, V. D., & Pietra, S. D. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*.

Elidan, G., McGraw, I., & Koller, D. (2006). Residual belief propagation: Informed scheduling for asynchronous message passing. *Proc. UAI*.

Finkel, J., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. *ACL*.

Freitag, D. (1998). *Machine learning for information extraction in informal domains*. Doctoral thesis, CMU.

Gould, S., Rodgers, J., Cohen, D., Elidan, G., & Koller, D. (2008). Multi-class segmentation with relative location prior. *International Journal on Computer Vision*.

Heskes, T. (2006). Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies. *Journal of Artificial Intelligence Research*, *26*, 153–190.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. ICML 18*.

McCallum, A., Freitag, D., & Pereira, F. C. N. (2000). Maximum entropy markov models for information extraction and segmentation. *Proc. ICML* (pp. 591–598).

Minka, T. P. (2001). Algorithms for maximum-likelihood logistic regression. Available from `http://www.stat.cmu.edu/~minka/`.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann.

Ren, X., & Malik, J. (2003). Learning a classification model for segmentation. *Proc. ICCV*.

Shental, N., Zomet, A., Hertz, T., & Weiss, Y. (2003). Learning and inferring image segmentations using the GBP typical cut algorithm. *Proc. ICCV*.

Sontag, D., & Jaakkola, T. (2007). New outer bounds on the marginal polytope. *Proc. NIPS 21*.

Sutton, C., & McCallum, A. (2004). Collective segmentation and labeling of distant entities in information extraction. *ICML workshop on Statistical Relational Learning*.

Sutton, C., & McCallum, A. (2005). Piecewise training of undirected models. *Proc. UAI*.

Sutton, C., & Minka, T. (2006). *Local training and belief propagation* (Technical Report). Microsoft Research.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. *Proc. UAI 18*.

Teh, Y. W., & Welling, M. (2001). The unified propagation and scaling algorithm. *Proc. NIPS*.

Teh, Y. W., & Welling, M. (2003). On improving the efficiency of the iterative proportional fitting procedure. *AISTATS*.

Vishwanathan, S. V. N., Schraudolph, N. N., Schmidt, M. W., & Murphy, K. P. (2006). Accelerated training of conditional random fields with stochastic gradient methods. *ICML '06* (pp. 969–976).

Wainwright, M., Jaakkola, T., & Willsky, A. (2003). Tree-reweighted belief propagation algorithms and approximate ML estimation via pseudo-moment matching. *Proc. AISTATS*.

Wainwright, M., & Jordan, M. (2006). Log=determinant relaxation for approximate inference in discrete Markov random fields. *IEEE Transactions on Signal Processing*.

Wainwright, M. J. (2006). Estimating the "wrong" graphical model: Benefits in the computation-limited setting. *J. Mach. Learn. Res.*, *7*, 1829–1859.

Weiss, Y., Yanover, C., & Meltzer, T. (2007). MAP estimation, linear programming and belief propagation with convex free energies. *Proc. UAI*.

Yedidia, J., Freeman, W., & Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transaction on Information Theory*, *51*, 2282–2312.

Yuille, A. L. (2002). CCCP algorithms to minimize the Bethe and Kikuchi free energies: convergent alternatives to belief propagation. *Neural Comput.*, *14*, 1691–1722.

Zhu, C., Byrd, R., Lu, P., & Nocedal, J. (1997). L-BFGS-B: Algorithm 778: FORTRAN subroutines for large-scale bound constrained optimization. *ACM Transactions on Mathematical Software*, *23*.

Multi-View Learning over Structured and Non-Identical Outputs

Kuzman Ganchev* **João V. Graça†** **John Blitzer‡** **Ben Taskar***

*Computer & Information Science
University of Pennsylvania
Philadelphia, PA

†L²F INESC-ID
INESC-ID
Lisboa, Portugal

‡Natural Language Computing
Microsoft Research Asia
Beijing, China

Abstract

In many machine learning problems, labeled training data is limited but unlabeled data is ample. Some of these problems have instances that can be factored into multiple views, each of which is nearly sufficent in determining the correct labels. In this paper we present a new algorithm for probabilistic multi-view learning which uses the idea of stochastic agreement between views as regularization. Our algorithm works on structured and unstructured problems and easily generalizes to partial agreement scenarios. For the full agreement case, our algorithm minimizes the Bhattacharyya distance between the models of each view, and performs better than CoBoosting and two-view Perceptron on several flat and structured classification problems.

1 Introduction

Multi-view learning refers to a set of semi-supervised methods which exploit redundant views of the same input data (Blum & Mitchell, 1998; Collins & Singer, 1999; Brefeld et al., 2005; Sindhwani et al., 2005). These multiple views can come in the form of context and spelling features in the case of text processing and segmentation, hypertext link text and document contents for document classification, and multiple cameras or microphones in the case of speech and vision. Multi-view methods typically begin by assuming that each view alone can yield a good predictor. Under this assumption, we can regularize the models from each view by constraining the amount by which we permit them to disagree on unlabeled instances. This regularization can lead to better convergence by significantly decreasing the effective size of our hypothesis class (Balcan & Blum, 2005; Kakade & Foster, 2007; Rosenberg & Bartlett, 2007).

In this paper, we propose a probabilistic agreement framework based on minimizing the Bhattacharyya distance (Kailath, 1967) between models trained using different views. Our regularizer is well-suited for optimization of the log-loss, and we give a fast optimization algorithm based on constrained EM (Graca et al., 2008). Where our work is most similar to co-regularization schemes, a minimum Kullbeck-Leibler (KL) distance projection can be computed in closed form resulting in an algorithm that performs better than both CoBoosting and two view Perceptron on several natural language processing tasks. In addition our framework allows us to use different training sets for the two classifiers, even if they have a different label set. In that case, we can reduce the hypothesis space by preferring pairs of models that agree on *compatible* labeling of unlabeled data rather than on *identical* labeling, while still minimizing KL in closed form. When the two views come from models that differ not only in the label set but also in the model structure of the output space, our framework can still encourage agreement, but the KL minimization cannot be computed in closed form. Finally, our method uses soft assignments to latent variables resulting in a more stable optimization procedure.

2 Stochastic Agreement

This section outlines stochastic agreement regularization and our method for optimizing it via constrained EM. We can generalize the discussion for multiple views, but for simplicity focus on two views here. We begin by considering the setting of complete agreement. In this setting we have a common desired output for the two models and we believe that each of the two views is sufficiently rich to predict labels accurately. We can leverage this knowledge by restricting our search to model pairs p_1,p_2 that satisfy

$p_1(y \mid x) \approx p_2(y \mid x)$. Our co-regularized objective is

$$\min_\theta \mathcal{L}_1(\theta_1) + \mathcal{L}_2(\theta_2) + c\mathbf{E}_U[B(p_1(\theta_1), p_2(\theta_2))] \quad (1)$$

where $\mathcal{L}_i = \mathbf{E}[-\log(p_i(y_i|x;\theta_i))] + \frac{1}{\sigma_i^2}||\theta_i||^2$ for $i = 1, 2$ are the standard regularized log likelihood losses of the models p_1 and p_2, $\mathbf{E}_U[B(p_1, p_2)]$ is the expected Bhattacharyya distance (Kailath, 1967) between the predictions of the two models on the unlabeled data, and c is a constant defining the relative weight of the unlabeled data. The Bhattacharyya distance between two distributions is given by

$$B(p_1, p_2) = -\log \sum_y \sqrt{p_1(y)p_2(y)}.$$

It is a very natural, symmetric measure of distance between distributions which has been used in many signal detection applications (Kailath, 1967). It is also related to the well-known Hellinger distance as well KL-divergence as we outline below.

2.1 Optimization algorithm

In order to optimize the objective we first consider a variational definition of Bhattacharyya distance that will allow us to generalize our approach to structured and non-identical output spaces. Proposition 2.1 relates the Bhattacharyya regularization term to the value of a constrained minimum KL problem, where the constraints enforce agreements between the two views. Proposition 2.2 shows that a simple majorization-minimziation algorithm optimizes the desired objective. The proof is in the appendix.

Proposition 2.1 *The Bhattacharyya distance* $-\log \sum_y \sqrt{p_1(y)p_2(y)}$ *is equal to* $\frac{1}{2}$ *of the value of the convex optimization problem*

$$\begin{aligned} \min_{q \in \mathcal{Q}} \quad & \mathrm{KL}(q(y_1, y_2)||p_1(y_1)p_2(y_2)) \\ \text{where} \quad & \mathcal{Q} = \{q : \mathbf{E}_q[\delta(y_1 = y) - \delta(y_2 = y)] = 0 \ \forall y\}, \end{aligned}$$

where $\delta(cond)$ *is* 1 *if cond is true and* 0 *otherwise. Furthermore, the minimizer decomposes as* $q(y_1, y_2) = q_1(y_1)q_2(y_2)$ *and is given by* $q_i(y) \propto \sqrt{p_1(y)p_2(y)}$.

Replacing the Bhattacharyya regularization term in Equation 1 with the program of Proposition 2.1 yields the objective

$$\min_\theta \mathcal{L}_1(\theta) + \mathcal{L}_2(\theta) + c\mathbf{E}_U \left[\min_{q \in \mathcal{Q}(\mathbf{x})} \mathrm{KL}(q(y_1 y_2) \,||\, p(y_1)p(y_2)) \right].$$

Note that this objective is convex separately in θ and q, but not jointly. We can optimize it iteratively using the constrained EM framework of Graca et al. (2008). We define $\mathbf{agree}(p_1, p_2)$ to be the minimizer of Proposition 2.1, and present the optimization in Algorithm 1.

Algorithm 1 minimizes co-regularized loss:

$$\mathcal{L}_1(\theta) + \mathcal{L}_2(\theta) + c\mathbf{E}_U[\min_{q \in \mathcal{Q}(\mathbf{x})} \mathrm{KL}(q(y_1, y_2) \,||\, p_1(y_1)p_2(y_2)] \,.$$

1: $\theta_1 \leftarrow \min_\theta \mathcal{L}_1(\theta_1)$
2: $\theta_2 \leftarrow \min_\theta \mathcal{L}_2(\theta_2)$
3: **for** n iterations **do**
4: $\quad q(y_1, y_2|\mathbf{x}) \leftarrow \mathbf{agree}(p_1(y_1|x), p_2(y_2|x)) \ \ \forall x \in U$
5: $\quad \theta_1 \leftarrow \min_\theta \ \mathcal{L}_1(\theta) - c \mathop{\mathbf{E}}_{x \sim U, y_1 \sim q}[\log p_1(y_1|x;\theta)]$
6: $\quad \theta_2 \leftarrow \min_\theta \ \mathcal{L}_2(\theta) - c \mathop{\mathbf{E}}_{x \sim U, y_2 \sim q}[\log p_2(y_2|x;\theta)]$
7: **end for**

Proposition 2.2 *Fixpoints of the loop in Algorithm 1 are local minima of*

$$\mathcal{L}_1(\theta_1) + \mathcal{L}_2(\theta_2) + c\mathbf{E}_U \left[\min_{q \in \mathcal{Q}(\mathbf{x})} \mathrm{KL}(q(y_1, y_2) \,||\, p(y_1)p(y_2)) \right].$$

If we consider the labels of the unlabeled data to be hidden variables, then Algorithm 1 is the constrained EM algorithm of Graca et al. (2008). Line 4 corresponds to the constrained E-step, while lines 5 and 6 constitute the M-step of the algorithm. The proof of Proposition 2.2 is a straightforward modification of the constrained EM proof in Graca et al. (2008).

Recall that we defined $\mathbf{agree}(p_1, p_2)$ to be the minimizer of the convex program in Proposition 2.1. The same proposition gives us a simple closed-form solution for this minimizer. In Section 2.2 shows that this solution is also easy to compute for structured models. We then show in Section 2.3 how a simple change to the constraints of the convex program allows us to optimize a natural co-regularizer in the partial agreement setting.

2.2 Undirected graphical models

Our regularizer extends to full agreement for undirected graphical models. In the case where p_1 and p_2 have the same structure, $q = \mathbf{agree}(p_1, p_2)$ will share this structure and the projection can be computed in closed form. Let $p_1(y \mid x) = Z_1^{-1} \prod_c \phi_1(y_c, x)$, where c is a clique in the graphical model and $\phi_1(y_c, x)$ is the corresponding clique potential, similarly for p_2. The derivation follows from the fact that $q(y_1, y_2)$ factors into a product of $q_1(y_1) = q_2(y_2)$, such that:

$$\begin{aligned} q_i(y)^2 &\propto p_1(y)p_2(y) && (2) \\ &= Z_1^{-1} Z_2^{-1} \prod_c \phi_1(y_c)\phi_2(y_c) && (3) \\ &= Z_q^{-2} \prod_c \phi_q(y_c)^2, && (4) \end{aligned}$$

where $\phi_q(y_c) = \sqrt{\phi_1(y_c)\phi_2(y_c)}$. This means that we can represent q using the same cliques as p_1 and p_2, and the clique potentials for q are just the square root of the product of the clique potentials for p_1 and p_2. In the case of log-linear Markov random fields, the clique potentials are stored in log space so this corresponds to averaging the values before computing normalization.

2.3 Partial agreement and hierarchical labels

Our method extends naturally to partial-agreement scenarios. For example we can encourage two part of speech taggers with different tag sets to produce compatible parts of speech, such as noun in tag set one and singular-noun in tag set 2 rather than noun in tag set 1 and verb in tag set 2. In particular, suppose we have a mapping from both label sets into a common space where it makes sense to encourage agreement. For the part of speech tagging example, this could mean mapping all nouns from both tagsets into a single class, all verbs into another class and so on. In general, we might have functions $g_1(y_1)$ and $g_2(y_2)$ that map variables for the two models onto the same space. We want to encourage agreement with $p_1(z) \approx p_2(z)$ where $p_1(z) = \sum_y \delta(g_1(y) = z)p_1(y)$ and similarly for p_2. Also, denote $p_i(y_i|z) = p_i(y_i)/p_i(z)$. In this case, our objective becomes

$$\min_\theta \mathcal{L}_1(\theta_1) + \mathcal{L}_2(\theta_2) + c\mathbf{E}_U[B(p_1(z), p_2(z))] \; . \quad (5)$$

In the special case where some labels are identical for the two models and others are incompatible, we have $g_1(y_1)$ mapping the incompatible labels into one bin and the others into their own special bins.

By slightly changing the constraints to the program in Proposition 2.1. With slight abuse of notation we denote $g_1(y_1) = z$ as $y_1 \mapsto z$. The following proposition allows us to optimize equation 5.

Proposition 2.3 *The Bhattacharyya distance* $-\log \sum_z \sqrt{p_1(z)p_2(z)}$ *is* $\frac{1}{2}$ *the value of the convex program*

$$\begin{array}{ll} \min\limits_{q \in \mathcal{Q}} & \mathrm{KL}(q(y_1, y_2)||p_1(y_1)p_2(y_2)) \\ \text{where} & \mathcal{Q} = \{q : \mathbf{E}_q[\delta(y_1 \mapsto z) - \delta(y_2 \mapsto z)] = 0 \; \forall z\} \, . \end{array}$$

Furthermore, the minimizer decomposes as $q(y_1, y_2) = q_1(y_1|z_1)q_1(z_1)q_2(y_2|z_2)q_2(z_2)$, *where* $q_1(z_1) = q_2(z_2) \propto \sqrt{p_1(z_1)p_2(z_2)}$ *and* $q_1(y_1|z_1) = p_1(y_1|z_1)$ *and* $q_2(y_2|z_2) = p_2(y_2|z_2)$.

By using the program in Proposition 2.3 instead of the one in Proposition 2.1 as the distribution $\mathbf{agree}(p_1, p_2)$ in Algorithm 1) we optimize the partial agreement objective using Algorithm 1. Note in particular that the computation of $\mathbf{agree}(p_1, p_2)$ is still in closed form.

2.4 Partial agreement in structured problems

For the unstructured case, it is easy to compute $p(z)$. Unfortunately if we collapse some labels, $p(z)$ might not have the same Markov properties as $p(y)$. For example, if p is a distribution over three states (1,2,3) that assigns probability 1 to the sequence (1,2,3,1,2...) and probability zero to other sequences. This is a first-order Markov chain. If the mapping is $1 \mapsto 1$ and $2, 3 \mapsto 0$ then $p(z)$ assigns probability 1 to (1,0,0,1,0...), which cannot be represented as a first-order Markov chain. Essentially, the original chain relied on being able to distinguish between the allowable transition (2,3) and the disallowed transition (3,2). When we collapse the states, both of these transitions map to (0,0) and cannot be distinguished. Consequently, the closed form solution given in Proposition 2.3 is not usable. Potentially, we could compute some approximation to $p(z)$ and from that compute an approximation to q. Instead, we re-formulate our constraints for the structured case and use conjugate gradient to find the optimal q. We ensure that the proposal distribution will have the same form as p_1 and p_2 by requiring only that the marginals of each clique match the desired marginals rather than requiring the joint to have the same probability:

$$\begin{array}{ll} \min\limits_{q} & \mathrm{KL}(q(y_1, y_2)||p_1(y_1)p_2(y_2)) \\ \text{s.t.} & \mathbf{E}_q[\delta(y_{1,c} \mapsto z_c) - \delta(y_{2,c} \mapsto z_c)] = 0 \; \forall z_c. \end{array} \quad (6)$$

In our experiments we found that 10 gradient steps are sufficient for practical purposes.

3 Relation to previous work

There has been a significant amount of work on two view learning in the past few years. Co-Training (Blum & Mitchell, 1998) labels increasingly large amounts of unlabeled data with a classifier from each view and re-trains each classifier on the unlabeled data. If the two views are generated independently of each other given the label, Co-Training is guaranteed to work. Nigam & Ghani (2000) find that this assumption is violated in practice and that Co-Training performance suffers significantly. Pierce & Cardie (2001) empirically investigate Co-Training for noun phrase chunking and find the same problem. They suggest correcting by hand the examples labeled by the classifiers. Unlike Co-Training, CoBoosting (Collins & Singer, 1999) and two view Perceptron (Brefeld et al., 2005) re-label the unlabeled data at each iteration. Potentially this might lead to a more stable optimization algorithm, since the identity of the first few labeled examples has a smaller effect on the outcome. Of course directly optimizing the objective would lead to even

more stability. Sindhwani et al. (2005) introduce co-regularized least squares and co-regularized SVMs and directly optimize the objective functions they propose. Like them, we know what our objective function is and our optimization procedure is guaranteed to find a local optimum. Kakade & Foster (2007) investigate two view linear regression and show that the hypothesis space can be reduced by performing CCA on input data.

In the case where we compute the KL projection in closed form, we are essentially using the two classifiers as a logarithmic opinion pool (Bordley, 1982). Smith et al. (2005) find that using logarithmic opinion pools of unregularized conditional random fields performs as well as regularized CRFs, and does not require tuning a regularization parameter. This suggests that co-regularized CRFs should be less sensitive to regularization hyper-parameters than a monolithic CRF. Suzuki et al. (2007) use a logarithmic opinion pool of several discriminatively trained CRFs and several hidden Markov models for named entity recognition and syntactic chunking. Computing optimal mixing weights on a held-out portion of the training set, they iteratively re-train the HMMs on the data labeled with the opinion pool. Since their optimization algorithm is very similar to ours, we can derive an objective function for their optimization from our work.

There is an entire class of different semi-supervised methods which assume that the true decision boundary lies on low density regions of the input space Chapelle et al. (2006). For example, Jiao et al. (2006) minimizes entropy on unlabeled data for CRFs. Mann & McCallum (2007) is also quite related, as it regularizes feature expectations on unlabeled data.

In the next section we empirically compare our method with CoBoosting and two view Perceptron. Since these methods are based on different objective functions it is worth examining where each one works best. Altun et al. (2003) compare log-loss and exp-loss for sequential problems. They find that the loss function does not have as great an effect on performance as the feature choice. However, they also note that exp-loss is expected to perform better for clean data, while log-loss is expected to perform better when there is label noise. The intuition behind this is in the rate of growth of the loss functions. Exp-loss grows exponentially with misclassification margin while log-loss grows linearly. Consequently when there is label noise, AdaBoost focuses more on modeling the noise. Since Co-Boosting optimizes a co-regularized exp-loss while our work optimizes a co-regularized log-loss we expect to do better on problems where the labels are noisy.

To get more intuition about this, Figure 1 shows the

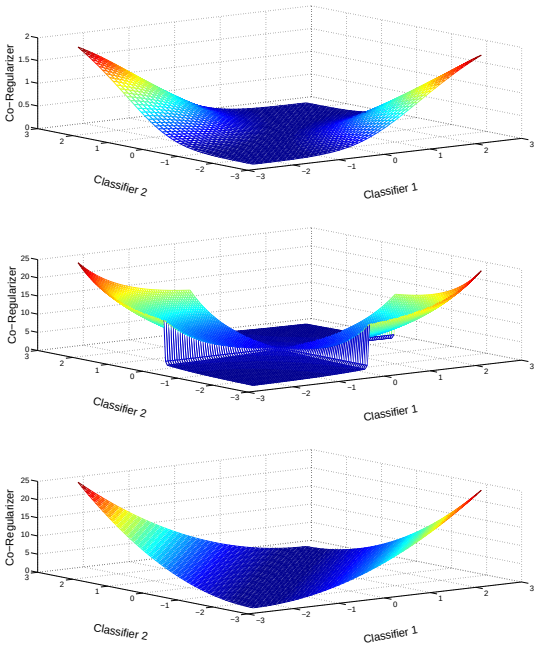


Figure 1: Different Loss Functions. Top: Bhattacharyya distance regularization. Middle: Exp-loss regularization. Bottom: Least Square regularization

co-regularization loss functions for our method (top), CoBoosting (middle) and co-RLS (bottom). For two underlying binary linear classifiers, $\hat{y} = \text{sign}(w \cdot x)$, the horizontal axes represent the value of the dot product, while the vertical axis is the loss. If we consider the plane parallel to the page, we see how the different co-regularizers penalize the classifiers when they disagree and are equally confident in their decision. Restricted to this plane, all three co-regularizers grow at the same asymptotic rate as the loss functions for the individual models: Linearly for our work, exponentially for Co-Boosting and quadratically for co-RLS. If we look at the area where the two models agree (the flat part of the Co-Boosting graph) we see what the penalty is when the classifiers agree but have different confidence. In this case co-RLS is harshest since it penalizes differences in the dot product equally regardless of the absolute value of the dot product. Intuitively, this is a problem. If one model predicts 1 with confidence 0.5 and the other predicts -1 with confidence 0.5 they are disagreeing while if they both predict 1 with confidence 1000 and 1001 respectively, they are agreeing on the label and are very close in their confidence estimates. At the other extreme, Co-Boosting imposes almost no penalty whenever the two classifiers agree, regardless of their confidence. The Bhattacharyya distance co-regularizer lies between these extremes, penalizing differences in confidence near the origin but is more lenient when the classifiers are both very confi-

dent and agree.

We compare our work with two view Perceptron for both structured and unstructured problems. Altun et al. (2002) describe an extension of AdaBoost to structured problems and an extension of CoBoosting is immediate, but we did not implement it due to time constraints.

Finally, if we have labeled data from one domain but want to apply it to another domain we can use any of the co-training frameworks mentioned earlier, including our own to perform domain transfer. For sentiment classification we will see that our method performs comparably with Structural Correspondence Learning (Blitzer et al., 2006), which is based on Alternating Structure Optimization (Ando & Zhang, 2005).

4 Experiments

4.1 Classification problems

Our first set of experiments is for transfer learning for sentiment classification. We used the data from Blitzer et al. (2007). The two views are generated from a random split of the features. We compare our method to several supervised methods as well as Co-Boosting (Collins & Singer, 1999), two view Perceptron (Brefeld et al., 2005) and structural correspondence learning (Blitzer et al., 2007). The column labeled "SCL" contains the best results from Blitzer et al. (2007), and is not directly comparable with the other methods since it uses some extra knowledge about the transfer task to choose auxiliary problems. For all the two-view methods we weigh the total labeled data equally with the total unlabeled data. We regularize the maximum entropy classifiers with a unit variance Gaussian prior. The results are in Table 1. Our method is labeled "SAR" (Stochastic Agreement Regularization). Out of the 12 transfer learning task, our method performs best in 6 cases, SCL in 4, while CoBoosting performs best only once. Two view Perceptron never outperforms all other methods. One important reason for the success of our method is the relative strength of the maximum entropy classifier relative to the other supervised methods for this particular task. We expect that Co-Boosting will perform better than our method in situations where Boosting significantly outperforms maximum entropy.

The next set of experiments we performed was with named entity disambiguation. Given a set of split named entities, we want to predict what type of named entity each one is. We use the training data from the 2003 CoNLL shared task (Sang & Meulder, 2003). The two views comprise content versus context features. The content features were words, POS tags and character n-grams of length 3 for all tokens in the named entity, while context features the same but for three words before and after the named entity. We used 2000 examples as testing data and roughly 30,000 as unlabeled data. Table 2 shows the results for different amounts of training data. For this dataset, we choose the variance of the Gaussian prior as well as the relative weighting of the labeled and unlabeled data by cross validation on the training set. In order to test whether the advantage our method gets is from the joint objective or from the use of $\mathbf{agree}(p_1, p_2)$, which is an instance of logarithmic opinion pools, we also report the performance of using $\mathbf{agree}(p_1, p_2)$ when the two views p_1 and p_2 have been trained only on the labeled data. In the column labeled "$\mathbf{agree}_0$" we see that for this dataset the benefit of our method comes from the joint objective function rather than from the use of logarithmic opinion pools.

Data size	mx-ent	$\mathbf{agree}_0$	SAR	RRE
500	74.0	74.4	76.4	9.2%
1000	80.0	80.0	81.7	8.5%
2000	83.4	83.4	84.8	8.4%

Table 2: Named entity disambiguation. Prior variance and c chosen by cross validation. $\mathbf{agree}_0$ refers to performance of two view model before first iteration of EM. RRE is reduction in error relative to error of MaxEnt model.

4.2 Partial Agreement

We also wanted to investigate the applicability of this method to the partial agreement setting. Suppose that you are interested in a classification task and you have labeled a small training corpus. You also have available a training corpus from some other source that has some of the distinctions you are interested in, but not others. You could use our method by training a classifier for your task as well as one for the auxiliary data and encouraging them to agree on the appropriate labels on unlabeled data. To investigate this scenario, we chose four newsgroups from the 20-newsgroups corpus (Lang, 1995), namely comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, talk.politics.mideast, rec.sport.baseball. Our views consist of one model trained on a subset of the newsgroups data with all four labels and another model trained on a subset where talk.politics.mideast and rec.sport.baseball have been collapsed into one category. Figure 2 shows the average accuracies over 10 runs with differing amounts of training data. In all cases, the number of labeled examples per view are equal and we regularize using a Gaussian prior with variance 10. The labeled and unlabeled data have equal weighting for these experiments. We see in the

Domains	MIRA	Boost	Perc	mx-ent	SCL	CoBoost	coPerc	SAR
books→dvds	77.2	72.0	74	78.5	75.8	78.8	75.5	**79.8**
dvds→books	72.8	74.8	74.5	80.3	79.7	79.8	74.5	**81.3**
books→electr	70.8	70.3	73.3	72.5	75.9	**77.0**	69.3	75.5
electr→books	70.7	62.5	73	72.8	**75.4**	71.0	67.5	74.3
books→kitchn	74.5	76.3	73.5	77.8	78.9	78.0	76.5	**81.0**
kitchn→books	70.9	66.5	67.3	70.3	68.6	69.8	66	**72.8**
dvds→electr	73.0	73.2	73.5	75.5	74.1	75.3	71.2	**76.5**
electr→dvds	70.6	66.3	64.8	69.3	**76.2**	73.5	63.3	73.0
dvds→kitchn	74.0	75.5	78.3	80.5	81.4	79.0	78.25	**82.8**
kitchn→dvds	72.7	61.8	64	69.5	**76.9**	70.1	60.5	72.8
electr→kitchn	84.0	73.2	81	**86.5**	85.9	85.0	83.3	85.8
kitchn→electr	82.7	66.3	81	82.8	**86.8**	83.0	80.5	85.5

Table 1: Performance of several methods on a sentiment classification transfer learning task. Reviews of objects of one type are used to train a classifier for reviews of objects of another type. The abbreviations in the column names are as follows. Boost: AdaBoost algorithm, Perc: Perceptron, mx-ent: maximum entropy, SCL: structural correspondence learning, CoBoost: CoBoosting, coPerc: two view Perceptron, SAR: this work. The best accuracy is shown in bold for each task.

figure that while SAR performs better than the supervised model, most of the benefit comes from agreement during inference (Agree 0 curve in Figure 2).

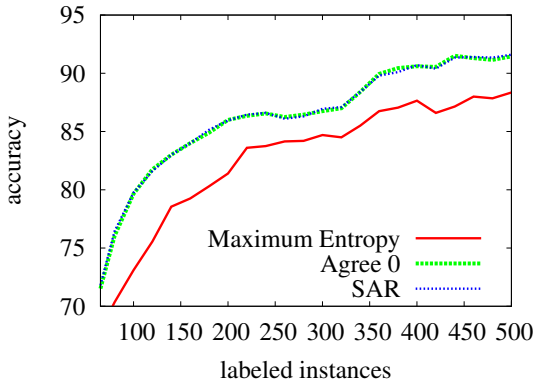


Figure 2: Results of the partial agreement experiments with four labels in one view and three labels in the other.

Table 3 shows the average confusion matrix for the runs with 500 training examples. Rows represents correct labels, while the columns represent labels assigned by the model. Each cell contains the percentage of all instances with the given correct label for the fully supervised case on top, and the difference to the same value for the partial agreement case below. We see that the auxiliary partially labeled data helps us to reduce the confusion between the difficult comp.sys.ibm.pc.hardware, comp.sys.mac.hardware categories. We see a similar trend in confusion matrices for all amounts of training data.

	pc	mac	pol	bball
pc	82.9	16.5	0	0.5
	5.1	**−5.6**	0.4	0.2
mac	15.0	83.7	0.2	1.2
	−3.7	**3.6**	**−0.2**	0.1
politics	1.4	2.3	93.2	3.2
	−0.7	**−1.2**	**1.6**	0.2
bball	1.9	2.5	0.6	95.1
	−0.7	**−1.9**	0.2	**2.2**

Table 3: Part of the confusion matrix for the partial agreement scenario. Top in each cell: Percents for fully supervised. Bottom in each cell: Percent difference to partial agreement. Bold represents improvement. See text for description.

4.3 Structured models

In order to investigate the applicability of our method to structured learning we apply it to the shallow parsing task of noun phrase chunking. We performed our experiments on the English training portion of the CoNLL 2000 shared task (Sang & Buchholz, 2000). We selected 500 sentences as testing data, varying amounts of data for training and the remainder was used as unlabeled data. We use content and context views, where the content view is the current word and POS tag while the context view is the previous and next words and POS tags. We regularize the CRFs with a variance 10 Gaussian prior and weigh the unlabeled data so that it has the same total weight as the labeled data. The variance value was chosen based on preliminary experiments with the data. Table 4 shows the F-1 scores of the different models. We compare our method to a monolithic CRF as well as av-

size	CRF	SAR(RRE)	Perc	coPerc
10	73.2	**78.2** (19%)	69.4	71.2
20	79.4	**84.2** (23%)	74.4	76.8
50	86.3	**86.9** (4%)	80.1	84.1
100	88.5	**88.9** (3%)	86.1	88.1
200	89.6	89.6 (0%)	89.3	**89.7**
500	**91.3**	90.6 (-8%)	90.8	90.9
1000	91.6	91.1 (-6%)	91.5	**91.8**

Table 4: F-1 scores for noun phrase chunking with context/content views. Testing data comprises 500 sentences, with 8436 sentences divided among training and unlabeled data. The best score is shown in bold for each training data size.

eraged Perceptron the two view Perceptron of Brefeld et al. (2005) with averaging. The Perceptron models were trained for 20 iterations. Preliminary experiments show that performance on held out data does not change after 10 iterations so we believe the models have converged. Both two view semi-supervised methods show gains over the corresponding fully-supervised method for 10-100 sentences of training data, but do not improve further as the amount of labeled data increases. The method presented in this paper outperforms two view Perceptron when the amount of labeled data is very small, probably because regularized CRFs perform better than Perceptron for small amounts of data. As the number of training sentences increases, two view Perceptron performs as well as our method, but at this point it has little or no improvement over the fully-supervised Perceptron.

4.4 Structured partial agreement

Finally, we ran some named entity recognition experiments where the two views had different label sets. All our experiments use the training portion of the 2003 CoNLL shared task (Sang & Meulder, 2003), regularize the model parameters with a variance 10 Gaussian prior and weigh the labeled and unlabeled data equally. The view of primary interest had segmentation of names in the categories "person", "location", "organization" and "miscellaneous". The auxiliary view collapsed the "location" and "miscellaneous" labels. For the experiments we use a total of about 14 thousand sentences, of which 200 are used for testing, and the rest are split between training for each of the two models and "unlabeled" data. The F-1 score as we vary the amount of training data available to each model are shown in Figure 3. The number of training sentences are the same for both views and are shown on the horizontal axis. We see that the semi-supervised model does better than the baseline in most cases, but most of the improvement is from the combination of the two classifiers rather than from joint learning.

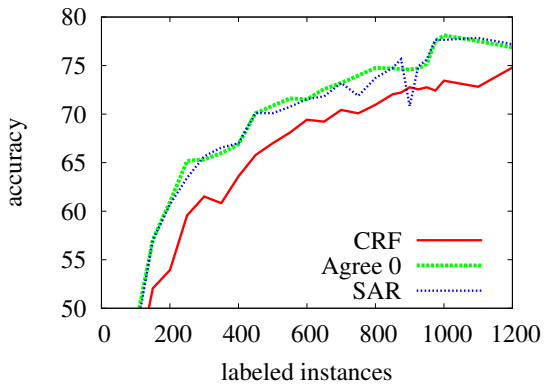


Figure 3: Results of the partial agreement experiments for structure output on name entity resolution, with all (four) categories in one view and three categories in the other (Location and Misc where collapsed into misc).

5 Conclustion and Future Work

We have introduced a novel two-view co-regularization appropriate for probabilistic models, which naturally extends to multiple views. In the normal two-view setting where the output spaces of the two views are identical, the co-regularization penalty is the Bhattacharyya distance. Our framework extends naturally to structured problems and also to partial agreement scenarios. We compared the framework with CoBoosting and two view Perceptron as well as a state of the art transfer learning algorithm, and found that our method often outperforms all other methods. Additionally, we demonstrate that our framework can be used in cases where the output spaces of the two views are not identical and the other methods are not directly applicable. A natural extension of this work would be to encourage agreement between more than two views, and to weigh the views differently as in logarithmic opinion pools. Another direction for future work is the application of this framework to problems where the proposal distribution cannot be computed in closed form, such as combining a dependency parser with a phrase structure parser. Finally, it would be interesting to investigate under what conditions two view co-regularization frameworks such as ours can be expected to work. For example, under what conditions should we expect the two models to converge asymptotically faster than a monolithic model?

Acknowledgements

Kuzman Ganchev was partially supported by NSF ITR EIA 0205448. João V. Graça was supported by a fellowship from Fundação para a Ciência e Tecnologia (SFRH/ BD/ 27528/ 2006).

References

Altun, Y., Hofmann, T., & Johnson, M. Discriminative learning for label sequences via boosting. In *NIPS 15*. MIT Press, 2002.

Altun, Y., Johnson, M., & Hofmann, T. Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proc. EMNLP*. ACL, 2003.

Ando, R. K. & Zhang, T. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:pp. 1817–1853, 2005.

Balcan, M. & Blum, A. A PAC-style model for learning from labeled and unlabeled data. In *Proc. COLT*. 2005.

Blitzer, J., Dredze, M., & Pereira, F. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. ACL*. ACL, 2007.

Blitzer, J., McDonald, R., & Pereira, F. Domain adaptation with structural correspondence learning. In *Proc. EMNLP*. 2006.

Blum, A. & Mitchell, T. Combining labeled and unlabeled data with co-training. In *COLT*. 1998.

Bordley, R. F. A multiplicative formula for aggregating probability assessments. *Management Science*, 28(10):pp. 1137–1148, 1982.

Brefeld, U., Büscher, C., & Scheffer, T. Multi-view hidden markov perceptrons. In *LWA*. 2005.

Chapelle, O., Schölkopf, B., & Zien, A., eds. *Semi-Supervised Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006.

Collins, M. & Singer, Y. Unsupervised models for named entity classification. In *Proc. SIGDAT-EMNLP*. ACL, 1999.

Graca, J., Ganchev, K., & Taskar, B. Expectation maximization and posterior constraints. In *NIPS 20*. MIT Press, 2008.

Jiao, F., Wang, S., Lee, C.-H., Greiner, R., & Schuurmans, D. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proc. ACL*, pp. 209–216. ACL, Morristown, NJ, USA, 2006.

Kailath, T. The divergence and bhattacharyya distance measures in signal selection. *IEEE Transactions on Communications*, 15(1):pp. 52–60, 1967.

Kakade, S. & Foster, D. Multi-view regression via canonical correlation analysis. In *Learning Theory*, vol. 4539. Springer Berlin / Heidelberg, 2007.

Lang, K. Newsweeder: Learning to filter netnews. In *Proc. ICML*. 1995.

Mann, G. S. & McCallum, A. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proc. ICML*. 2007.

Nigam, K. & Ghani, R. Understanding the behavior of co-training. In *Proc. workshop on text mining of KDD-2000*. 2000.

Pierce, D. & Cardie, C. Limitations of co-training for natural language learning from large datasets. In *Proc. EMNLP*. 2001.

Rosenberg, D. & Bartlett, P. The rademacher complexity of co-regularized kernel classes. In M. Meila & X. Shen, eds., *Proc. AI Stats*. 2007.

Sang, E. F. T. K. & Buchholz, S. Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. CoNLL-2000 and LLL-2000*. 2000.

Sang, E. F. T. K. & Meulder, F. D. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proc. HLT-NAACL*. ACL, 2003.

Sindhwani, V., Niyogi, P., & Belkin, M. A Co-Regularization Approach to Semi-supervised Learning with Multiple Views. In *Workshop on Learning with Multiple Views, Proceedings of ICML*. 2005.

Smith, A., Cohn, T., & Osborne, M. Logarithmic opinion pools for conditional random fields. In *Proc. ACL*. ACL, 2005.

Suzuki, J., Fujino, A., & Isozaki, H. Semi-supervised structured output learning based on a hybrid generative and discriminative approach. In *Proc. EMNLP-CoNLL*. 2007.

A Proof of proposition 2.1

Taking the dual of the optimization problem in Equation 2.1 we get

$$\arg\max_{\lambda} -\log \sum_{y_1,y_2} p(y_1,y_2)\exp(\lambda\cdot\psi) \qquad (7)$$

with $q(y_1,y_2) \propto p(y_1,y_2)\exp(\lambda\cdot\psi(y_1,y_2))$. Where $\psi(y_1,y_2)$ is a vector of features of the form $\delta(y_1 = y) - \delta(y_2 = y)$ with one entry for each possible label y. Noting that the features decompose into $\psi'(y_1) - \psi'(y_2)$, we know that $q(y_1,y_2)$ decomposes as $q_1(y_1)q_2(y_2)$. Furthermore, our constraints require that $q_1(y) = q_2(y)\forall y$ so we have $q(y_1)q(y_2) \propto p_1(y_1)\exp(\lambda\cdot\psi'(y_1))p_2(y_2)\exp(-\lambda\cdot\psi'(y_2))$. Letting $y_1 = y_2$ we have $q(y)^2 = p_1(y)p_2(y)$ which gives us a closed form computation of $\mathbf{agree}(p_1,p_2) \propto \sqrt{p_1(y)p_2(y)}$. Substituting this solution into the program of Proposition 2.1, and performing algebraic simplification yields the desired result.

AND/OR Importance Sampling

Vibhav Gogate and Rina Dechter
Department of Information and Computer Science,
University of California, Irvine, CA 92697,
{vgogate,dechter}@ics.uci.edu

Abstract

The paper introduces AND/OR importance sampling for probabilistic graphical models. In contrast to importance sampling, AND/OR importance sampling caches samples in the AND/OR space and then extracts a new sample mean from the stored samples. We prove that AND/OR importance sampling may have lower variance than importance sampling; thereby providing a theoretical justification for preferring it over importance sampling. Our empirical evaluation demonstrates that AND/OR importance sampling is far more accurate than importance sampling in many cases.

1 Introduction

Many problems in graphical models such as computing the probability of evidence in Bayesian networks, solution counting in constraint networks and computing the partition function in Markov random fields are *summation problems*, defined as a sum of a function over a domain. Because these problems are NP-hard, sampling based techniques are often used to approximate the sum. The focus of the current paper is on *importance sampling*.

The main idea in importance sampling [Geweke, 1989, Rubinstein, 1981] is to transform the summation problem to that of computing a weighted average over the domain by using a special distribution called the proposal (or importance) distribution. Importance sampling then generates samples from the proposal distribution and approximates the true average over the domain by an average over the samples; often referred to as the sample average. The sample average is simply a ratio of the sum of sample weights and the number of samples, and it can be computed in a *memory-less* fashion since it requires keeping only these two quantities in memory.

The main idea in this paper is to equip importance sampling with memoization or caching in order to exploit conditional independencies that exist in the graphical model. Specifically, we cache the samples on an AND/OR tree or graph [Dechter and Mateescu, 2007] which respects the structure of the graphical model and then compute a new weighted average over that AND/OR structure, yielding, as we show, an unbiased estimator that has a smaller variance than the importance sampling estimator. Similar to AND/OR search [Dechter and Mateescu, 2007], our new AND/OR importance sampling scheme recursively combines samples that are cached in independent components yielding an increase in the effective sample size which is part of the reason that its estimates have lower variance.

We present a detailed experimental evaluation comparing importance sampling with AND/OR importance sampling on Bayesian network benchmarks. We observe that the latter outperforms the former on most benchmarks and in some cases quite significantly.

The rest of the paper is organized as follows. In the next section, we describe preliminaries on graphical models, importance sampling and AND/OR search spaces. In sections 3, 4 and 5 we formally describe AND/OR importance sampling and prove that its sample mean has lower variance than conventional importance sampling. Experimental results are described in section 6 and we conclude with a discussion of related work and summary in section 7.

2 Preliminaries

We represent sets by bold capital letters and members of a set by capital letters. An assignment of a value to a variable is denoted by a small letter while bold small letters indicate an assignment to a set of variables.

Definition 2.1 (belief networks). A *belief network (BN)* is a graphical model $\mathscr{R} = (\mathbf{X}, \mathbf{D}, \mathbf{P})$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of random variables over multi-valued domains $\mathbf{D} = \{\mathbf{D_1}, \ldots, \mathbf{D_n}\}$. Given a directed acyclic graph G over $\mathbf{X}$, $\mathbf{P} = \{P_i\}$, where $P_i = P(X_i|\mathbf{pa}(X_i))$ are conditional probability tables (CPTs) associated with each X_i. $\mathbf{pa}(X_i)$ is the set of parents of the variable X_i in G. A belief network represents a probability distribution over $\mathbf{X}$, $P(\mathbf{X}) =$

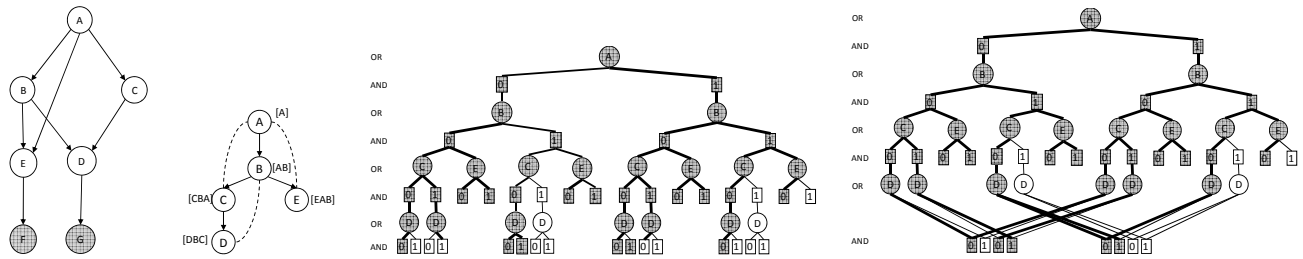


Figure 1: (a) Bayesian Network, (b) Pseudo-tree (c) AND/OR tree (d) AND/OR search graph

$\prod_{i=1}^{n} P(X_i|\mathbf{pa}(X_i))$. An *evidence set* $\mathbf{E}=\mathbf{e}$ is an instantiated subset of variables.The moral graph (or primal graph) of a belief network is the undirected graph obtained by connecting the parent nodes and removing direction.

Definition 2.2 (**Probability of Evidence).** Given a belief network $\mathscr{R}$ and evidence $\mathbf{E}=\mathbf{e}$, the probability of evidence $P(\mathbf{E}=\mathbf{e})$ is defined as:

$$P(\mathbf{e}) = \sum_{\mathbf{X}\setminus\mathbf{E}} \prod_{j=1}^{n} P(X_j|\mathbf{pa}(X_j))_{|\mathbf{E}=\mathbf{e}} \tag{1}$$

The notation $h(\mathbf{X})_{|\mathbf{E}=\mathbf{e}}$ stands for a function h over $\mathbf{X}\setminus\mathbf{E}$ with the assignment $\mathbf{E}=\mathbf{e}$.

2.1 AND/OR search spaces

We can compute probability of evidence by search, by accumulating probabilities over the search space of instantiated variables. In the simplest case, this process defines an OR search tree, whose nodes represent partial variable assignments. This search space does not capture the structure of the underlying graphical model. To remedy this problem, [Dechter and Mateescu, 2007] introduced the notion of AND/OR search space. Given a bayesian network $\mathscr{R}=(\mathbf{X},\mathbf{D},\mathbf{P})$, its AND/OR search space is driven by a pseudo tree defined below.

Definition 2.3 (**Pseudo Tree).** Given an undirected graph $G=(V,E)$, a directed rooted tree $T=(V,E)$ defined on all its nodes is called pseudo tree if any arc of G which is not included in E is a back-arc, namely it connects a node to an ancestor in T.

Definition 2.4 (**Labeled AND/OR tree).** Given a graphical model $\mathscr{R}=\langle\mathbf{X},\mathbf{D},\mathbf{P}\rangle$, its primal graph G and a backbone pseudo tree T of G, the associated AND/OR search tree, has alternating levels of AND and OR nodes. The OR nodes are labeled X_i and correspond to the variables. The AND nodes are labeled $\langle X_i,x_i\rangle$ and correspond to the value assignments in the domains of the variables. The structure of the AND/OR search tree is based on the underlying backbone tree T. The root of the AND/OR search tree is an OR node labeled by the root of T.

Each OR arc, emanating from an OR node to an AND node is associated with a **label** which can be derived from the CPTs of the bayesian network [Dechter and Mateescu, 2007]. Each OR node and AND node is also associated with a **value** that is used for computing the quantity of interest.

Semantically, the OR states represent alternative assignments, whereas the AND states represent problem decomposition into independent subproblems, all of which need be solved. When the pseudo-tree is a chain, the AND/OR search tree coincides with the regular OR search tree. The probability of evidence can be computed from a labeled AND/OR tree by recursively computing the value of all nodes from leaves to the root [Dechter and Mateescu, 2007].

Example 2.5. Figure 1(a) shows a bayesian network over seven variables with domains of $\{0,1\}$. F and G are evidence nodes. Figure 1(c) shows the AND/OR-search tree for the bayesian network based on the Pseudo-tree in Figure 1(b). Note that because F and G are instantiated, the search space has only 5 variables.

2.2 Computing Probability of Evidence Using Importance Sampling

Importance sampling [Rubinstein, 1981] is a simulation technique commonly used to evaluate the sum, $M=\sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x})$ for some real function f. The idea is to generate samples $\mathbf{x^1},\ldots,\mathbf{x^N}$ from a proposal distribution Q (satisfying $f(\mathbf{x})>0 \Rightarrow Q(\mathbf{x})>0$) and then estimate M as follows:

$$M = \sum_{\mathbf{x}\in\mathbf{X}} f(\mathbf{x}) = \sum_{\mathbf{x}\in\mathbf{X}} \frac{f(\mathbf{x})}{Q(\mathbf{x})} Q(\mathbf{x}) = \mathbb{E}_Q[\frac{f(\mathbf{x})}{Q(\mathbf{x})}] \tag{2}$$

$$\widehat{M} = \frac{1}{N}\sum_{i=1}^{N} w(\mathbf{x}^i)\ ,\ \textit{where}\ \ w(\mathbf{x}^i) = \frac{f(\mathbf{x}^i)}{Q(\mathbf{x}^i)} \tag{3}$$

w is often referred to as the sample weight. It is known that the expected value $\mathbb{E}(\widehat{M})=M$ [Rubinstein, 1981].

To compute the probability of evidence by importance sampling, we use the substitution:

$$f(\mathbf{x}) = \prod_{j=1}^{n} P(X_j|\mathbf{pa}(X_j))_{|\mathbf{E}=\mathbf{e}} \tag{4}$$

Several choices are available for the proposal distribution $Q(\mathbf{x})$ ranging from the prior distribution as in likelihood weighting to more sophisticated alternatives such as

IJGP-Sampling [Gogate and Dechter, 2005] and EPIS-BN [Yuan and Druzdzel, 2006] where the output of belief propagation is used to compute the proposal distribution.

As in prior work [Cheng and Druzdzel, 2000], we assume that the proposal distribution is expressed in a factored product form: $Q(\mathbf{X}) = \prod_{i=1}^{n} Q_i(X_i|X_1,\ldots,X_{i-1}) = \prod_{i=1}^{n} Q_i(X_i|\mathbf{Y_i})$, where $\mathbf{Y_i} \subseteq \{X_1,\ldots,X_{i-1}\}$, $Q_i(X_i|\mathbf{Y_i}) = Q(X_i|X_1,\ldots,X_{i-1})$ and $|\mathbf{Y_i}| < c$ for some constant c. We can generate a full sample from Q as follows. For i = 1 to n, sample $X_i = x_i$ from the conditional distribution $Q(X_i|X_1 = x_1,\ldots,X_{i-1} = x_{i-1})$ and set $X_i = x_i$.

3 AND/OR importance sampling

We first discuss computing expectation by parts; which forms the backbone of AND/OR importance sampling. We then present the AND/OR importance sampling scheme formally and derive its properties.

3.1 Estimating Expectation by Parts

In Equation 2, the expectation of a multi-variable function is computed by summing over the entire domain. This method is clearly inefficient because it does not take into account the decomposition of the multi-variable function as we illustrate below.

Consider the tree graphical model given in Figure 2(a). Let $A = a$ and $B = b$ be the evidence variables. Let $Q(ZXY) = Q(Z)Q(X|Z)Q(Y|Z)$ be the proposal distribution. For simplicity, let us assume that $f(Z) = P(Z)$, $f(XZ) = P(Z|X)P(A = a|X)$ and $f(YZ) = P(Z|Y)P(B = b|Y)$. We can express probability of evidence $P(a,b)$ as:

$$P(a,b) = \sum_{XYZ} \frac{f(Z)f(XZ)f(YZ)}{Q(Z)Q(X|Z)Q(Y|Z)} Q(Z)Q(X|Z)Q(Y|Z) = \mathbb{E}\left[\frac{f(Z)f(XZ)f(YZ)}{Q(Z)Q(X|Z)Q(Y|Z)}\right] \quad (5)$$

We can decompose the expectation in Equation 5 into smaller components as follows:

$$P(a,b) = \sum_{Z} \frac{f(Z)Q(Z)}{Q(Z)} \left(\sum_{X} \frac{f(XZ)Q(X|Z)}{Q(X|Z)}\right)\left(\sum_{Y} \frac{f(YZ)Q(Y|Z)}{Q(Y|Z)}\right) \quad (6)$$

The quantities in the two brackets in Equation 6 are, by definition, conditional expectations of a function over X and Y respectively given Z. Therefore, Equation 6 can be written as:

$$P(a,b) = \sum_{Z} \frac{f(Z)}{Q(Z)} \mathbb{E}\left[\frac{f(XZ)}{Q(X|Z)}|Z\right] \mathbb{E}\left[\frac{f(YZ)}{Q(Y|Z)}|Z\right] Q(Z) \quad (7)$$

By definition, Equation 7 can be written as:

$$P(a,b) = \mathbb{E}\left[\frac{f(Z)}{Q(Z)} \mathbb{E}\left[\frac{f(XZ)}{Q(X|Z)}|Z\right] \mathbb{E}\left[\frac{f(YZ)}{Q(Y|Z)}|Z\right]\right] \quad (8)$$

We will refer to Equations of the form 8 as *expectation by parts* borrowing from similar terms such as integration and summation by parts. If the domain size of all variables is $d = 3$, for example, computing expectation using Equation 5 would require summing over $d^3 = 3^3 = 27$ terms while computing the same expectation by parts would require summing over $d + d^2 + d^2 = 3 + 3^2 + 3^2 = 21$ terms. Therefore, exactly computing expectation by parts is clearly more efficient.

Importance sampling ignores the decomposition of expectation while approximating it by the sample average. Our new algorithm estimates the true expectation by decomposing it into several conditional expectations and then approximating each by an appropriate weighted average over the samples. Since computing expectation by parts is less complex than computing expectation by summing over the domain; we expect that approximating it by parts will be easier as well. We next illustrate how to estimate expectation by parts on our example Bayesian network given in Figure 2(a).

Assume that we are given samples $(z^1,x^1,y^1),\ldots,(z^N,x^N,y^N)$ generated from Q decomposed according to Figure 2(a). For simplicity, let $\{0,1\}$ be the domain of Z and let $Z = 0$ and $Z = 1$ be sampled N_0 and N_1 times respectively. We can approximate $\mathbb{E}\left[\frac{f(XZ)}{Q(X|Z)}|Z\right]$ and $\mathbb{E}\left[\frac{f(YZ)}{Q(Y|Z)}|Z\right]$ by $\widehat{g_X(Z=j)}$ and $\widehat{g_Y(Z=j)}$ defined below:

$$\widehat{g_X(Z=j)} = \frac{1}{N_j}\sum_{i=1}^{N} \frac{f(x^i,Z=j)I(x^i,Z=j)}{Q(x^i,Z=j)}$$
$$\widehat{g_Y(Z=j)} = \frac{1}{N_j}\sum_{i=1}^{N} \frac{f(y^i,Z=j)I(y^i,Z=j)}{Q(y^i,Z=j)} \quad (9)$$

where $I(x^i,Z=j)$ (or $I(y^i,Z=j)$) is an indicator function which is 1 iff the tuple $(x^i,Z=j)$ (or $(y^i,Z=j)$) is generated in any of the N samples and 0 otherwise.

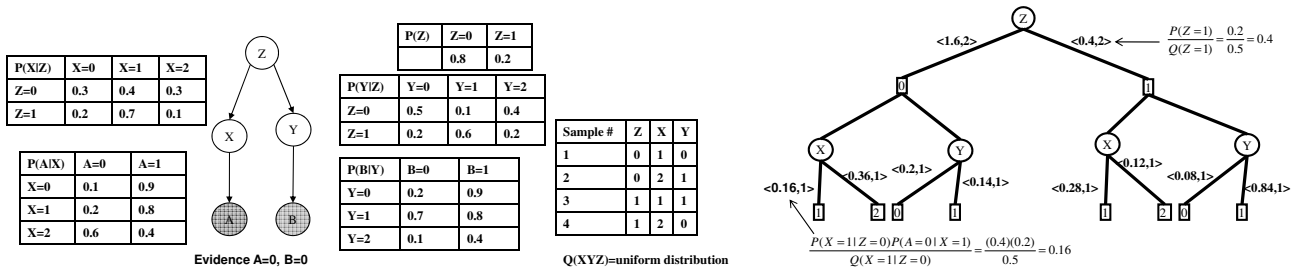
From Equation 8, we can now derive the following unbiased estimator for $P(a,b)$:

$$\widehat{P(a,b)} = \frac{1}{N}\sum_{j=0}^{1} \frac{N_j f(Z=j)\widehat{g_X(Z=j)}\widehat{g_Y(Z=j)}}{Q(Z=j)} \quad (10)$$

Importance sampling on the other hand would estimate $P(a,b)$ as follows:

$$\widetilde{P(a,b)} = \frac{1}{N}\sum_{j=0}^{1} N_j \frac{f(Z=j)}{Q(Z=j)} \times \frac{1}{N_j}\sum_{i=1}^{N} \frac{f(x^i,Z=j)f(y^i,Z=j)}{Q(x^i|Z=j)Q(Y^i|Z=j)} I(x^i,y^i,Z=j) \quad (11)$$

where $I(x^i,y^i,Z=j)$ is an indicator function which is 1 iff the tuple $(x^i,y^i,Z=j)$ is generated in any of the N samples and 0 otherwise.



(a) (b) (c)

Figure 2: (a) Bayesian Network, its CPTs, (b) Proposal Distribution and Samples (c) AND/OR sample tree

Equation 10 which is an unbiased estimator of expectation by parts given in Equation 8 provides another rationale for preferring it over the usual importance sampling estimator given by Equation 11. In particular in Equation 10, we estimate two functions defined over the random variables $X|Z=z$ and $Y|Z=z$ respectively from the generated samples. In importance sampling, on the other hand, we estimate a function over the joint random variable $XY|Z=z$ using the generated samples. Because the samples for $X|Z=z$ and $Y|Z=z$ are considered independently in Equation 10, N_j samples drawn over the joint random variable $XY|Z=z$ in Equation 11 correspond to a larger set $N_j * N_j = N_j^2$ of virtual samples. We know that [Rubinstein, 1981] the variance (and therefore the mean-squared error) of an unbiased estimator decreases with an increase in the effective sample size. Consequently, our new estimation technique will have lower error than the conventional approach.

In the following subsection, we discuss how the AND/OR structure can be used for estimating expectation by parts yielding the *AND/OR importance sampling* scheme.

3.2 Computing Sample Mean in AND/OR-space

In this subsection, we formalize the ideas of estimating expectation by parts on a general AND/OR tree starting with some required definitions. We define the notion of an AND/OR sample tree which is restricted to the generated samples and which will be used to compute the AND/OR sample mean. The labels on this AND/OR tree are set to account for the importance weights.

Definition 3.1 (Arc Labeled AND/OR Sample Tree). Given a a graphical model $\mathscr{R} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P} \rangle$, a pseudo-tree $T(V,E)$, a proposal distribution $Q = \prod_{i=1}^{n} Q(X_i|\mathbf{Anc}(X_i))$ such that $\mathbf{Anc}(\mathbf{X_i})$ is a subset of all ancestors of X_i in T, a sequence of assignments (samples) $\mathbf{S}$ and a complete AND/OR search tree ϕ_T, an AND/OR sample tree S_{AOT} is constructed from ϕ_T by removing all edges and corresponding nodes which are not in $\mathbf{S}$ i.e. they are not sampled.

The Arc-label for an OR node X_i to an AND node $X_i = x_i$ in S_{AOT} is a pair $\langle w, \# \rangle$ where:

- $w = \frac{P(X_i=x_i, \mathbf{anc}(x_i))}{Q(X_i=x_i|\mathbf{anc}(x_i))}$ is called the weight of the arc. $\mathbf{anc}(x_i)$ is the assignment of values to all variables from the node X_i to the root node of S_{AO} and $P(X_i = x_i, \mathbf{anc}(x_i))$ is the product of all functions in $\mathscr{R}$ that mention X_i but do not mention any variable ordered below it in T given $(X_i = x_i, \mathbf{anc}(x_i))$.
- # is the frequency of the arc. Namely, it is equal to the number of times the assignment $(X_i = x_i, \mathbf{anc}(x_i))$ is sampled.

Example 3.2. Consider again the Bayesian network given in Figure 2(a). Assume that the proposal distribution $Q(XYZ)$ is uniform. Figure 2(b) shows four hypothetical random samples drawn from Q. Figure 2(c) shows the AND/OR sample tree over the four samples. Each arc from an OR node to an AND node in the AND/OR sample tree is labeled with appropriate frequencies and weights according to Definition 3.1. Figure 2(c) shows the derivation of arc-weights for two arcs.

The main virtue of arranging the samples on an AND/OR sample tree is that we can exploit the independencies to define the *AND/OR sample mean*.

Definition 3.3 (AND/OR Sample Mean). Given a AND/OR sample tree with arcs labeled according to Definition 3.1, the **value** of a node is defined recursively as follows. The value of leaf AND nodes is "1" and the value of leaf OR nodes is "0". Let $\mathbf{C}(\mathbf{n})$ denote the child nodes and $v(n)$ denotes the value of node n. If n is a AND node then: $v(n) = \prod_{n' \in \mathbf{C}(\mathbf{n})} v(n')$ and if n is a OR node then

$$v(n) = \frac{\sum_{n' \in \mathbf{C}(\mathbf{n})} (\#(n,n') w(n,n') v(n'))}{\sum_{n' \in \mathbf{C}(\mathbf{n})} \#(n,n')}$$

The **AND/OR sample mean** is the value of the root node.

We can show that the value of an OR node is equal to an unbiased estimate of the conditional expectation of the variable at the OR node given an assignment from the root to the parent of the OR node. Since all variables, except the evidence variables are unassigned at the root node, the value of the root node equals the AND/OR sample mean which is an unbiased estimate of probability of evidence. Formally,

THEOREM **3.4.** *The AND/OR sample mean is an unbiased estimate of probability of evidence.*

Example 3.5. The calculations involved in computing the sample mean on the AND/OR sample tree on our example

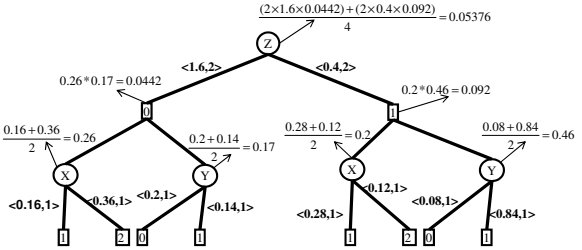


Figure 3: Computation of Values of OR and AND nodes in a AND/OR sample tree. The value of root node is equal to the AND/OR sample mean

Bayesian network given in Figure 2 are shown in Figure 3. Each AND node and OR node in Figure 3 is marked with a value that is computed recursively using definition 3.3. The value of OR nodes X and Y given $Z = j \in \{0,1\}$ is equal to $\widehat{g_X(Z=j)}$ and $\widehat{g_Y(Z=j)}$ respectively defined in Equation 9. The value of the root node is equal to the AND/OR sample mean which is equal to the sample mean computed by parts in Equation 10.

Algorithm 1 AND/OR Importance Sampling

Input: an ordering $O = (X_1, \ldots, X_n)$,a Bayesian network BN and a proposal distribution Q
Output: Estimate of Probability of Evidence

1: Generate samples $\mathbf{x}^1, \ldots, \mathbf{x}^N$ from Q along O.
2: Build a AND/OR sample tree S_{AOT} for the samples $\mathbf{x}^1, \ldots, \mathbf{x}^N$ along the ordering O.
3: Initialize all labeling functions $\langle w, \# \rangle$ on each arc from an Or-node n to an And-node n' using Definition 3.1.
4: **FOR** all leaf nodes i of S_{AOT} do
5: **IF And-node** v(i)= 1 **ELSE** v(i)=0
6: **For** every node n from leaves to the root do
7: Let $C(n)$ denote the child nodes of node n
8: **IF** $n = \langle X, x \rangle$ is a AND node, then $v(n) = \prod_{n' \in C(n)} v(n')$
9: **ELSE** if $n = X$ is a OR node then

$$v(n) = \frac{\sum_{n' \in C(n)} (\#(n,n') w(n,n') v(n'))}{\sum_{n' \in C(n)} \#(n,n')}.$$

10: Return v(root node)

We now have the necessary definitions to formally present the AND/OR importance sampling scheme (see Algorithm 1). In Steps 1-3, the algorithm generates samples from Q and stores them on an AND/OR sample tree. The algorithm then computes the AND/OR sample mean over the AND/OR sample tree recursively from leaves to the root in Steps $4-9$. We can show that the value $v(n)$ of a node in the AND/OR sample tree stores the sample average of the subproblem rooted at n, subject to the current variable instantiation along the path from the root to n. If n is the root, then $v(n)$ is the AND/OR sample mean which is our AND/OR estimator of probability of evidence. Finally, we summarize the complexity of computing AND/OR sample mean in the following theorem:

THEOREM **3.6.** *Given N samples and n variables (with constant domain size), the time complexity of computing AND/OR sample mean is $O(nN)$ (same as importance sampling) and its space complexity is $O(nN)$ (the space complexity of importance sampling is constant).*

4 Variance Reduction

In this section, we prove that the AND/OR sample mean may have lower variance than the sample mean computed using importance sampling (Equation 3).

THEOREM **4.1** (Variance Reduction)**.** *Variance of AND/OR sample mean is less than or equal to the variance of importance sampling sample mean.*

Proof. The details of the proof are quite complicated and therefore we only provide the intuitions involved. As noted earlier the guiding principle of AND/OR sample mean is to take advantage of conditional independence in the graphical model. Let us assume that we have three random variables $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ with the following relationship: $\mathbf{X}$ and $\mathbf{Y}$ are independent of each other given $\mathbf{Z}$ (similar to our example Bayesian network). The expression for variance derived here can be used in an induction step (induction is carried on the nodes of the pseudo tree) to prove the theorem.

In this case, importance sampling generates samples $((\mathbf{x}^1, \mathbf{y}^1, \mathbf{z}^1), \ldots, (\mathbf{x}^N, \mathbf{y}^N, \mathbf{z}^N))$ along the order $\langle \mathbf{Z}, \mathbf{X}, \mathbf{Y} \rangle$ and estimates the mean as follows:

$$\mu^{IS}(\mathbf{XYZ}) = \frac{\sum_{i=1}^{N} \mathbf{x}^i \mathbf{y}^i \mathbf{z}^i}{N} \tag{12}$$

Without loss of generality, let $\{\mathbf{z_1}, \mathbf{z_2}\}$ be the domain of $\mathbf{Z}$ and let these values be sampled N_1 and N_2 times respectively. We can rewrite Equation 12 as follows:

$$\mu^{IS}(\mathbf{XYZ}) = \frac{1}{N} \sum_{j=1}^{2} N_j \mathbf{z_j} \frac{\sum_{i=1}^{N} \mathbf{x}^i \mathbf{y}^i I(\mathbf{z}_j, \mathbf{x}^i, \mathbf{y}^i)}{N_j} \tag{13}$$

where $I(\mathbf{z}_j, \mathbf{x}^i, \mathbf{y}^i)$ is an indicator function which is 1 iff the partial assignment $(\mathbf{z}_j, \mathbf{x}^i, \mathbf{y}^i)$ is generated in any of the N samples and 0 otherwise.

AND/OR sample mean is defined as:

$$\mu^{AO}(\mathbf{XYZ}) = \frac{1}{N} \sum_{j=1}^{2} N_j \mathbf{z}_j \left(\frac{\sum_{i=1}^{N} \mathbf{x}^i I(\mathbf{z}_j, \mathbf{x}^i)}{N_j} \right) \left(\frac{\sum_{i=1}^{N} \mathbf{y}^i I(\mathbf{z}_j, \mathbf{y}^i)}{N_j} \right) \tag{14}$$

where $I(\mathbf{x}^j, \mathbf{z}_i)$ (and similarly $I(\mathbf{y}^j, \mathbf{z}_i)$) is an indicator function which equals 1 when one of the N samples contains the tuple $(\mathbf{x}^j, \mathbf{z}_i)$ (and similarly $(\mathbf{y}^j, \mathbf{z}_i)$)) and is 0 otherwise.

By simple algebraic manipulations, we can prove that the variance of estimator $\mu^{IS}(\mathbf{XYZ})$ is given by:

$$Var(\mu^{IS}(\mathbf{XYZ})) = \left(\sum_{j=1}^{2} \mathbf{z}_j^2 Q(\mathbf{z_j}) \Big(\mu(\mathbf{X}|\mathbf{z}_j)^2 V(\mathbf{Y}|\mathbf{z}_j) + \right.$$

$$\left. \mu(\mathbf{Y}|\mathbf{z}_j)^2 V(\mathbf{X}|\mathbf{z}_j) + V(\mathbf{X}|\mathbf{z}_j) V(\mathbf{Y}|\mathbf{z_j}) \Big) \right) / N - \mu^2_{\mathbf{XYZ}} / N \tag{15}$$

Similarly, the variance of AND/OR sample mean is given by:

$$Var(\mu^{AO}(\mathbf{XYZ})) = \left(\sum_{j=1}^{2} \mathbf{z}_j^2 Q(\mathbf{z_j}) \Big(\mu(\mathbf{X}|\mathbf{z}_j)^2 V(\mathbf{Y}|\mathbf{z}_j) + \mu(\mathbf{Y}|\mathbf{z}_j)^2 V(\mathbf{X}|\mathbf{z}_j) + \frac{V(\mathbf{X}|\mathbf{z}_j)V(\mathbf{Y}|\mathbf{z_j})}{N_j} \Big) \right) / N - \mu^2_{\mathbf{XYZ}}/N \quad (16)$$

where $\mu(\mathbf{X}|\mathbf{z}_j)$ and $V(\mathbf{X}|\mathbf{z}_j)$ are the conditional mean and variance respectively of $\mathbf{X}$ given $\mathbf{Z} = \mathbf{z}_j$. Similarly, $\mu(\mathbf{Y}|\mathbf{z}_j)$ and $V(\mathbf{Y}|\mathbf{z}_j)$ are the conditional mean and variance respectively of $\mathbf{Y}$ given $\mathbf{Z} = \mathbf{z}_j$.

From Equations 15 and 16, if $N_j = 1$ for all j, then we can see that the $Var(\mu^{AO}(\mathbf{XYZ})) = Var(\mu^{IS}(\mathbf{XYZ}))$. However if $N_j > 1$, $Var(\mu^{AO}(\mathbf{XYZ})) < Var(\mu^{IS}(\mathbf{XYZ}))$. This proves that the variance of AND/OR sample mean is less than or equal to the variance of conventional sample mean on this special case. As noted earlier using this case in induction over the nodes of a general pseudo-tree completes the proof.

□

5 Estimation in AND/OR graphs

Next, we describe a more powerful algorithm for estimating mean in AND/OR-space by moving from AND/OR-trees to AND/OR graphs as presented in [Dechter and Mateescu, 2007]. An AND/OR-tree may contain nodes that root identical subtrees. When such unifiable nodes are merged, the tree becomes a graph and its size becomes smaller. Some unifiable nodes can be identified using contexts defined below.

Definition 5.1 (**Context).** Given a belief network and the corresponding AND/OR search tree S_{AOT} relative to a pseudo-tree T, the context of any AND node $\langle X_i, x_i \rangle \in S_{AOT}$, denoted by $context(X_i)$, is defined as the set of ancestors of X_i in T, that are connected to X_i and descendants of X_i.

The context minimal AND/OR graph is obtained by merging all the context unifiable AND nodes. The size of the largest context is bounded by the tree width w^* of the pseudo-tree [Dechter and Mateescu, 2007]. Therefore, the time and space complexity of a search algorithm traversing the context-minimal AND/OR graph is $O(exp(w^*))$.

Example 5.2. For illustration, consider the context-minimal graph in Figure 1(e) of the pseudo-tree from Figure 1(c). Its size is far smaller that that of the AND/OR tree from Figure 2(c) (30 nodes vs. 38 nodes). The contexts of the nodes can be read from the pseudo-tree in Figure 1(b) as follows: context(A) = {A}, context(B) = {B,A}, context(C) = {C,B,A}, context(D) = {D,C,B} and context(E) = {E,A,B}.

The main idea in AND/OR-graph estimation is to store all samples on an AND/OR-graph instead of an AND/OR-tree. Similar to an AND/OR sample tree, we can define an identical notion of an AND/OR sample graph.

Definition 5.3 (**Arc labeled AND/OR sample graph).** Given a complete AND/OR graph ϕ_G and a set of samples $\mathbf{S}$, an AND/OR sample graph S_{AOG} is obtained by removing all nodes and arcs not in $\mathbf{S}$ from ϕ_G. The labels on S_{AOG} are set similar to that of an AND/OR sample tree (see Definition 3.1).

Example 5.4. The bold edges and nodes in Figure 1(c) define an AND/OR sample tree. The bold edges and nodes in Figure 1(d) define an AND/OR sample graph corresponding to the same samples that define the AND/OR sample tree in Figure 1(c).

The algorithm for computing the sample mean on AND/OR sample graphs is identical to the algorithm for AND/OR-tree (Steps 4-10 of Algorithm 1). The main reason in moving from trees to graphs is that the variance of the sample mean computed on an AND/OR sample graph can be even smaller than that computed on an AND/OR sample tree. More formally,

THEOREM **5.5.** *Let $V(\mu_{AOG})$, $V(\mu_{AOT})$ and $V(\mu_{IS})$ be the variance of AND/OR sample mean on an AND/OR sample graph, variance of AND/OR sample mean on an AND/OR sample tree and variance of sample mean of importance sampling respectively. Then given the same set of input samples:*

$$V(\mu_{AOG}) \leq V(\mu_{AOT}) \leq V(\mu_{IS})$$

We omit the proof due to lack of space.

THEOREM **5.6** (Complexity of computing AND/OR graph sample mean)**.** *Given a graphical model with n variables, a psuedo-tree with treewidth w^* and N samples, the time complexity of AND/OR graph sampling is $O(nNw^*)$ while its space complexity is $O(nN)$.*

6 Experimental Evaluation

6.1 Competing Algorithms

The performance of importance sampling based algorithms is highly dependent on the proposal distribution [Cheng and Druzdzel, 2000]. It was shown that computing the proposal distribution from the output of a Generalized Belief Propagation scheme of Iterative Join Graph Propagation (IJGP) yields better empirical performance than other available choices [Gogate and Dechter, 2005]. Therefore, we use the output of IJGP to compute the proposal distribution Q. The complexity of IJGP is time and space exponential in its i-bound, a parameter that bounds cluster sizes. We use a i-bound of 5 in all our experiments.

We experimented with three sampling algorithms for benchmarks which do not have determinism: (a) (pure) IJGP-sampling, (b) AND/OR-tree IJGP-sampling and (c) AND/OR-graph IJGP-sampling. Note that the underlying scheme for generating the samples is identical in all the

methods. What changes is the method of accumulating the samples and deriving the estimates. On benchmarks which have zero probabilities or determinism, we use the SampleSearch scheme introduced by [Gogate and Dechter, 2007] to overcome the rejection problem. We experiment with the following versions of SampleSearch on deterministic networks: (a) pure SampleSearch, (b) AND/OR-tree SampleSearch and (c) AND/OR-graph SampleSearch.

6.1.1 Results

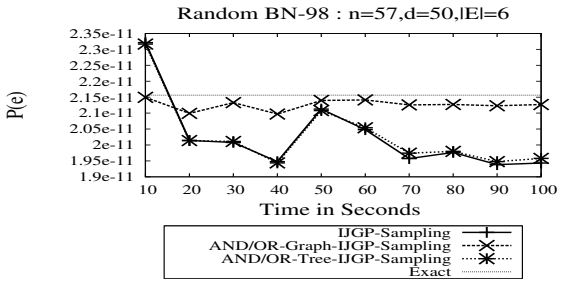
We experimented with three sets of benchmark belief networks (a) Random networks, (b) Linkage networks and (c) Grid networks. Note that only linkage and grid networks have zero probabilities on which we use SampleSearch.The exact $P(\mathbf{e})$ for most instances is available from the UAI 2006 competition web-site.

Our results are presented in Figures 4-6. Each Figure shows approximate probability of evidence as a function of time. The bold line in each Figure indicates the exact probability of evidence. The reader can visualize the error from the distance between the approximate curves and the exact line. For lack of space, we show only part of our results. Each Figure shows the number of variables n, the maximum-domain size d and the number of evidence nodes $|E|$ for the respective benchmark.

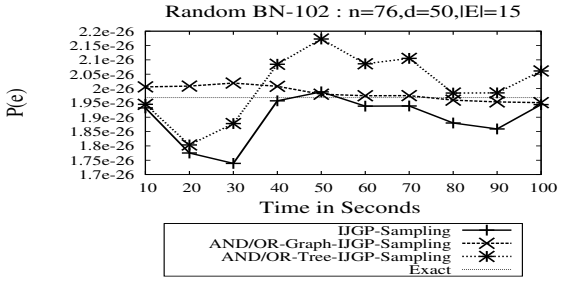
Random Networks From Figures 4(a) and 4(b), we see that AND/OR-graph sampling is better than AND/OR-tree sampling which in turn is better than pure IJGP-sampling. However there is not much difference in the error because the proposal distribution seems to be a very good approximation of the posterior.

Grid Networks All Grid instances have 1444 binary nodes and between 5-10 evidence nodes. From Figures 5(a) and 5(b), we can see that AND/OR-graph SampleSearch and AND/OR-tree SampleSearch are substantially better than pure SampleSearch.

Linkage Networks The linkage instances are generated by converting a Pedigree to a Bayesian network [Fishelson and Geiger, 2003]. These networks have between 777-2315 nodes with a maximum domain size of 36. Note that it is hard to compute exact probability of evidence in these networks [Fishelson and Geiger, 2003]. We observe from Figures 6(a),(b) (c) and (d) that AND/OR-graph SampleSearch is substantially more accurate than AND/OR-tree SampleSearch which in turn is substantially more accurate than pure SampleSearch. Notice the log-scale in Figures 6 (a)-(d) which means that there is an order of magnitude difference between the errors. Our results suggest that AND/OR-graph and tree estimators yield far better performance than conventional estimators especially on problems in which the proposal distribution is a bad approximation of the posterior distribution.

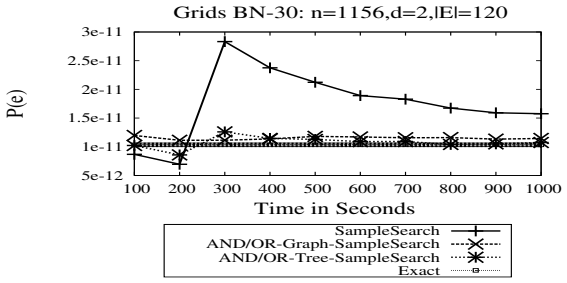


(a)

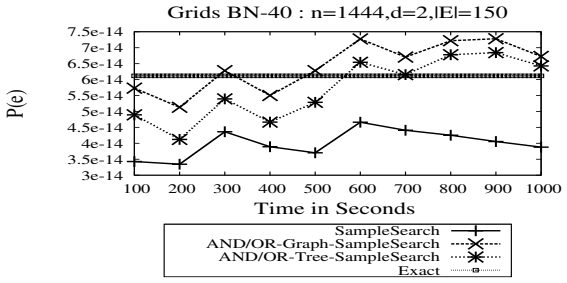


(b)

Figure 4: Random Networks



(a)

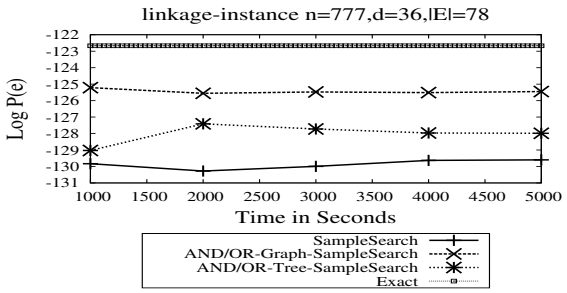


(b)

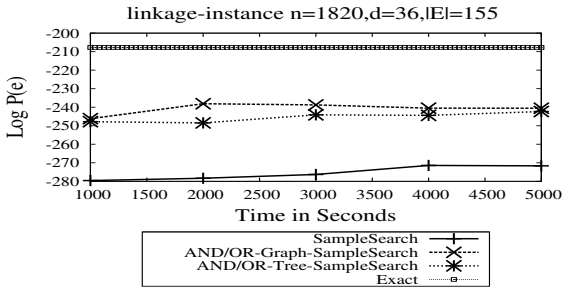
Figure 5: Grid Networks

7 Related Work and Summary

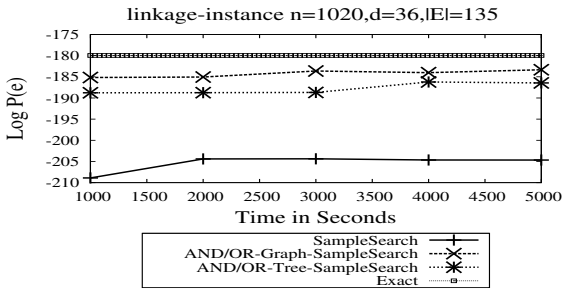
The work presented in this paper is related to the work by [Hernndez and Moral, 1995, Kjærulff, 1995, Dawid et al., 1994] who perform sampling based inference on a junction tree. The main idea in these papers is to perform message passing on a junction tree by substituting messages which are too hard to compute exactly by their sampling-based approx-



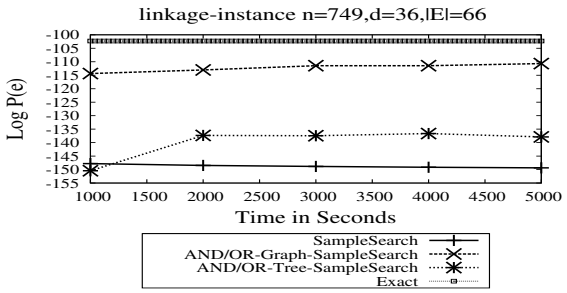
(a)



(b)



(c)



(d)

Figure 6: Linkage Bayesian Networks

imations. [Kjærulff, 1995, Dawid et al., 1994] use Gibbs sampling while [Hernndez and Moral, 1995] use importance sampling to approximate the messages. Similar to recent work on Rao-Blackwellised sampling such as [Bidyuk and Dechter, 2003, Paskin, 2004, Gogate and Dechter, 2005], variance reduction is achieved in these junction tree based sampling schemes because of some exact computations; as dictated by the Rao-Blackwell theorem. AND/OR estimation, however, does not require exact computations to achieve variance reduction. In fact, variance reduction due to Rao-Blackwellisation is orthogonal to the variance reduction achieved by AND/OR estimation and therefore the two could be combined to achieve more variance reduction. Also, unlike our work which focuses on probability of evidence, the focus of these aforementioned papers was on belief updating.

To summarize, the paper introduces a new sampling based estimation technique called AND/OR importance sampling. The main idea of our new scheme is to derive statistics on the generated samples by using an AND/OR tree or graph that takes advantage of the independencies present in the graphical model. We proved that the sample mean computed on an AND/OR tree or graph may have smaller variance than the sample mean computed using the conventional approach. Our experimental evaluation is preliminary but quite promising showing that on most instances AND/OR sample mean has lower error than importance sampling and sometimes by significant margins.

Acknowledgements

This work was supported in part by the NSF under award numbers IIS-0331707, IIS-0412854 and IIS-0713118 and the NIH grant R01-HG004175-02.

References

[Bidyuk and Dechter, 2003] Bidyuk, B. and Dechter, R. (2003). An empirical study of w-cutset sampling for bayesian networks. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*.

[Cheng and Druzdzel, 2000] Cheng, J. and Druzdzel, M. J. (2000). Ais-bn: An adaptive importance sampling algorithm for evidential reasoning in large bayesian networks. *J. Artif. Intell. Res. (JAIR)*, 13:155–188.

[Dawid et al., 1994] Dawid, A. P., Kjaerulff, U., and Lauritzen, S. L. (1994). Hybrid propagation in junction trees. In *IPMU'94*, pages 87–97.

[Dechter and Mateescu, 2007] Dechter, R. and Mateescu, R. (2007). AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2-3):73–106.

[Fishelson and Geiger, 2003] Fishelson, M. and Geiger, D. (2003). Optimizing exact genetic linkage computations. In *RECOMB 2003*.

[Geweke, 1989] Geweke, J. (1989). Bayesian inference in econometric models using monte carlo integration. *Econometrica*, 57(6):1317–39.

[Gogate and Dechter, 2005] Gogate, V. and Dechter, R. (2005). Approximate inference algorithms for hybrid bayesian networks with discrete constraints. *UAI-2005*.

[Gogate and Dechter, 2007] Gogate, V. and Dechter, R. (2007). Samplesearch: A scheme that searches for consistent samples. *AISTATS 2007*.

[Hernndez and Moral, 1995] Hernndez, L. D. and Moral, S. (1995). Mixing exact and importance sampling propagation algorithms in dependence graphs. *International Journal of Approximate Reasoning*, 12(8):553–576.

[Kjærulff, 1995] Kjærulff, U. (1995). Hugs: Combining exact inference and gibbs sampling in junction trees. In *UAI*, pages 368–375.

[Paskin, 2004] Paskin, M. A. (2004). Sample propagation. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

[Rubinstein, 1981] Rubinstein, R. Y. (1981). *Simulation and the Monte Carlo Method*. John Wiley & Sons, Inc., New York, NY, USA.

[Yuan and Druzdzel, 2006] Yuan, C. and Druzdzel, M. J. (2006). Importance sampling algorithms for Bayesian networks: Principles and performance. *Mathematical and Computer Modelling*.

Church: a language for generative models

"The Pope" Noah D. Goodman*, "The Infidel" Vikash K. Mansinghka*
"Altar Boy" Daniel Roy, "Brother" Keith Bonawitz & "Reverend" Joshua B. Tenenbaum
MIT BCS/CSAIL
Cambridge, MA 02139

Abstract

Formal languages for probabilistic modeling enable re-use, modularity, and descriptive clarity, and can foster generic inference techniques. We introduce *Church*, a universal language for describing stochastic generative processes. Church is based on the Lisp model of lambda calculus, containing a pure Lisp as its deterministic subset. The semantics of Church is defined in terms of evaluation histories and conditional distributions on such histories. Church also includes a novel language construct, the stochastic memoizer, which enables simple description of many complex non-parametric models. We illustrate language features through several examples, including: a generalized Bayes net in which parameters cluster over trials, infinite PCFGs, planning by inference, and various non-parametric clustering models. Finally, we show how to implement query on any Church program, exactly and approximately, using Monte Carlo techniques.

1 INTRODUCTION

Probabilistic models have proven to be an enormously useful tool in artificial intelligence, machine learning, and cognitive science. Most often these models are specified in a combination of natural and mathematical language, and inference for each new model is implemented by hand. Stochastic programming languages [e.g. 12, 14, 10] aim to tame the model-building process by giving a formal language which provides simple, uniform, and re-usable descriptions of a wide class of models, and supports generic inference techniques. In this paper we present the Church stochastic programming language (named for computation pioneer Alonzo Church), a universal language for describing generative processes and conditional queries over them. Because this language is based on Church's lambda calculus, expressions, which represent generative models, may be arbitrarily composed and abstracted. The distinctive features of Church, and the main contributions of this paper, are: 1) a Lisp-like language specification in which we view *evaluation as sampling* and *query as conditional sampling*, 2) a *stochastic memoizer*, which allows separate evaluations to share generative history and enables easy description of non-parametric probabilistic models, and, 3) generic schemes for exact and approximate inference, which implement the query primitive, so that any Church program may be run without writing special-purpose inference code.

2 THE CHURCH LANGUAGE

The Church language is based upon a pure subset of the functional language Scheme [6], a Lisp dialect. Church is a dynamically-typed, applicative-order language, in which procedures are first-class and expressions are values. Church expressions describe generative processes: the meaning of an expression is specified through a primitive procedure `eval`, which samples from the process, and a primitive procedure `query`, which generalizes `eval` to sample conditionally. In true Lisp spirit, `eval` and `query` are ordinary procedures that may be nested within a Church program. Randomness is introduced through stochastic primitive functions; memoization allows random computations to be reused.

Church expressions have the form:

$$\begin{aligned}\text{expression} \quad ::= \quad & c \mid x \mid (e_1\ e_2\ ...) \mid (\texttt{lambda}\ (x...)\ e) \mid \\ & (\texttt{if}\ e_1\ e_2\ e_3) \mid (\texttt{define}\ x\ e) \mid (\texttt{quote}\ e)\end{aligned}$$

Here x stands for a variable (from a countable set of

*The first two authors contributed equally to this work.

variable symbols), e_i for expressions, and c for a (primitive) constant. (We often write `'`e as shorthand for `(quote` e`)`.)

The constants include primitive data types (nil, Boolean, char, integer, fixed-precision real, etc.), and standard functions to build data structures (notably `pair`, `first`, and `rest` for lists) and manipulate basic types (e.g. `and`, `not`)[1]. As in most programming languages, all primitive types are countable; real numbers are approximated by either fixed- or floating-precision arithmetic. A number of standard (deterministic) functions, such as the higher-order function `map`, are provided as a standard library, automatically defined in the global environment. Other standard Scheme constructs are provided—such as `(let ((a` *a-def*`) (b` *b-def*`) ...)` *body*`)`, which introduces names that can be used in *body*, and is sugar for nested `lambda`s.

Church *values* include Church expressions, and procedures; if $v_1 ... v_n$ are Church values the list $(v_1 ... v_n)$ is a Church value. A Church *environment* is a list of pairs consisting of a variable symbol and a value (the variable is *bound* to the value); note that an environment is a Church value. Procedures come in two types: *Ordinary procedures* are triples, (`body`, `args`, `env`), of a Church expression (the body), a list of variable symbols (the formal parameters, or arguments), and an environment. *Elementary random procedures* are ordinary procedures that also have a *distribution function*—a probability function that reports the probability $P($ value $|$ `env`, `args`$)$ of a return value from evaluating the body (via the `eval` procedure described below) given `env` and values of the formal parameters[2].

To provide an initial set of elementary random procedures we allow stochastic primitive functions, in addition to the usual constants, that randomly sample a return value depending only on the current environment. Unlike other constants, these random functions are available only wrapped into elementary random procedures: `(fun, args, env, dist)`, where `dist` $= P($ value $|$ `env`, `args`$)$ is the probability function for `fun`. We include several elementary random procedures, such as `flip` which flips a fair coin (or flips a weighted coin when called with a weight argument).

[1]The primitive function `gensym` deserves special note: `(eval '(gensym) env)` returns a procedure (`c, x, env`) where `c` is a constant function which returns `True` if `x` is bound to the procedure (`c, x, env`), and `False` otherwise. Furthermore it is guaranteed that `(gensym (gensym))` evaluates to `False` (i.e. each evaluation of `gensym` results in a unique value).

[2]This definition implies that when the body of an elementary random procedure is not a constant, its distribution function represents the marginal probability over any other random choices made in evaluating the body. This becomes important for implementing query.

- `(eval '`c `env)`: For constant c, return c(`env`).
- `(eval '`x `env)`: Look-up symbol x in `env`, return the value it is bound to.
- `(eval '(`e_1 e_2 `...) env)`: Evaluate each `(eval '`e_i `env)`. The value of `(eval '`e_1 `env)` should be a procedure (`body`, x_2 `...`, `env2`). Make `env3` by extending `env2`, binding x_2 `...` to the return values of e_2 `...`. Return the value of `(eval body env3)`.
- `(eval '(lambda (`x...`)` e`) env)`: Return the procedure (e, x..., `env`).
- `(eval '(if` e_1 e_2 e_3`) env)`: If `(eval` e_1 `env)` returns `True` return the return value of `(eval` e_2 `env)`, otherwise of `(eval` e_3 `env)`.
- `(eval '(quote` e`) env)`: Return the expression e (as a value).
- `(eval '(define` x e`) env)`: Extend `env` by binding the value of `(eval '`e `env)` to x; return the extended environment.

Figure 1: An informal definition of the `eval` procedure. If preconditions of these descriptions fail the constant value `error` is returned. Note that constants represent (possibly stochastic) functions from environments to values—truly "constant" constants return themselves.

A Church expression defines a generative process via the recursive evaluation procedure, `eval`. This primitive procedure takes an expression and an environment and returns a value—it is an environment model, shared with Scheme, of Church's lambda calculus [4, 6]. The evaluation rules are given in Fig. 1. An evaluation history for an expression e is the sequence of recursive calls to `eval`, and their return values, made by `(eval '`e `env)`. The probability of a finite evaluation history is the product of the probabilities for each elementary random procedure evaluation in this history[3]. The *weight* of an expression in a particular environment is the sum of the probabilities of all of its finite evaluation histories. An expression is *admissible* in an environment if it has weight one, and a procedure is admissible if its body is admissible in its environment for all values of its arguments. An admissible expression defines a distribution on evaluation histories (we make this claim precise in section 2.2). Note that an admissible expression can have infinite histories, but the set of infinite histories must have probability zero. Thus admissibility can be thought of as the requirement that evaluation of an expression halts with probability one. Marginalizing this distribution over histories results in a distribution on values, which we write μ(`e, env`). Thus, `(eval '`e `env)`, for admissible e, returns a sample from μ(`e, env`).

[3]However, if evaluating an elementary random procedure results in evaluating another elementary random procedure we take only the probability of the first, since it already includes the second.

The procedure `eval` allows us to interpret Church as a language for generative processes, but for useful probabilistic inference we must be able to sample from a distribution conditioned on some assertions (for instance the posterior probability of a hypothesis conditioned on observed data). The procedure `(query 'e p env)` is defined to be a procedure which samples a value from μ(`e, env`) conditioned on the predicate procedure p returning `True` when applied to the value of `(eval 'e env)`. The environment argument `env` is optional, defaulting to the current environment. (Note that the special case of `query` when the predicate p is the constant procedure `(lambda (x) True)` defines the same distribution on values as `eval`.) For example, one might write `(query '(pair (flip) (flip)) (lambda (v) (+ (first v) (last v))))` to describe the conditional distribution of two flips given that at least one flip landed heads. If e or p are not admissible in `env` the query result is undefined. We describe this conditional distribution, and conditions for its well-definedness, more formally in Theorem 2.3. In Section 4 we consider Monte Carlo techniques for implementing `query`.

It can be awkward in practice to write programs using `query`, because many random values must be explicitly passed from the query expression to the predicate through the return value. An alternative is to provide a means to name random values which are shared by all evaluations, building up a "random world" within the query. To enable a this style of programming, we provide the procedure `lex-query` (for "lexicalizing query") which has the form:

```
(lex-query
  '((A A-definition)
    (B B-definition)
    ...)
  'e 'p)
```

where the first argument binds a lexicon of symbols to definitions, which are available in the environment in which the remaining (query and predicate) expressions are evaluated. In this form the predicate is an expression, and the final environment argument is omitted—the current environment is used.

A program in Church consists of a sequence of Church expressions—this sequence is called the *top level*. Any definitions at the top level are treated as extending the global (i.e. initial) environment, which then is used to evaluate the remaining top-level expressions. For instance:
`(define A` e_1`)` e_2
is treated as:
`(eval '`e_2 `(eval '(define A` e_1`) global-env))`.

2.1 Stochastic Memoization

In deterministic computation, memoization is a technique for efficient implementation that does not affect the language semantics: the first time a (purely functional) procedure is evaluated with given arguments its return value is recorded; thereafter evaluations of that procedure with those arguments directly return this value, without re-evaluating the procedure body. Memoization of a stochastic program can radically change the semantics: if `flip` is an ordinary random procedure `(= (flip) (flip))` is `True` with probability 0.5, but if `flip` is memoized this expression is `True` with probability one. More generally, a collection of memoized functions has a random-world semantics as discussed in [10]. In Section 3 we use memoization together with `lex-query` to describe generative processes involving an unknown number of objects with persistent features, similar to the BLOG language [12].

To formally define memoization in Church, we imagine extending the notion of environment to allow countably many variables to be bound in an environment. The higher-order procedure `mem` takes an admissible-procedure and returns another procedure: if `(eval` e `env)` returns the admissible procedure `(body, args, env2)`, then `(eval '(mem` e`) env)` returns the *memoized procedure* (`mfun`$_e$`, args, env+`), where:

- `env+` is `env2` (notionally) extended with a symbol $\mathtt{V}_{val}$, for each value *val*, bound to a value drawn from the distribution μ(`(`e *val*`), env`).
- `mfun`$_e$ is a new constant function such that `mfun`$_e$ applied to the environment `env+` extended with `args` bound to *val* returns the value bound to $\mathtt{V}_{val}$.

This definition implies that infinitely many random choices may be made when a memoized random procedure is created—the notion of admissibility must be extended to expressions which involve `mem`. In the next section we describe an appropriate extension of admissibility, such that admissible expressions still define a marginal distribution on values, and the conditional distributions defining `query` are well-formed.

Ordinary memoization becomes a semantically meaningful construct within stochastic languages. This suggests that there may be useful generalizations of `mem`, which are not apparent in non-stochastic computation. Indeed, instead of always returning the initial value or always re-evaluating, one could stochastically decide on each evaluation whether to use a previously computed value or evaluate anew. We define such a stochastic memoizer `DPmem` by using the Dirichlet process (DP) [20]—a distribution on discrete distributions

```
(define (DP alpha proc)
  (let ((sticks (mem (lambda x (beta 1.0 alpha))))
        (atoms  (mem (lambda x (proc)))))
    (lambda () (atoms (pick-a-stick sticks 1)))))

(define (pick-a-stick sticks J)
  (if (< (random) (sticks J))
      J
      (pick-a-stick sticks (+ J 1))))

(define (DPmem alpha proc)
  (let ((dps (mem (lambda args
                    (DP alpha
                        (lambda () (apply proc args))
                        )))))
    (lambda argsin ((apply dps argsin))) ))
```

Figure 2: Church implementation of the Dirichlet Process, via stick breaking, and `DPmem`. (Evaluating `(apply proc args)` in `env` for `args`=`(`$\mathtt{a}_1$ `...)` is equivalent to `(eval '(proc` $\mathtt{a}_1$ `...) env)`.)

built from an underlying base measure. For an admissible procedure `e`, the expression `(DPmem` a e`)` evaluates in `env` to a procedure which samples from a (fixed) sample from the DP with base measure μ(`e, env`) and concentration parameter a. (When a=0, `DPmem` reduces to `mem`, when a=∞, it reduces to the identity.) The notion of using the Dirichlet process to cache generative histories was first suggested in Johnson et al. [5], in the context of grammar learning. In Fig. 2 we write the Dirichlet Process and `DPmem` directly in Church, via a stick-breaking representation. This gives a definition of these objects, proves that they are semantically well-formed (provided the rest of the language is), and gives one possible implementation.

We pause here to explain choices made in the language definition. Programs written with pure functions, those that always return the same value when applied to the same arguments, have a number of advantages. It is clear that a random function cannot be pure, yet there should be an appropriate generalization of purity which maintains some locality of information. We believe the right notion of purity in a stochastic language is *exchangeability*: if an expression is evaluated several times in the same environment, the distribution on return values is invariant to the order of evaluations. This exchangeability is exploited by the Metropolis-Hastings algorithm for approximating `query` given in Section 4.

Mutable state (or an unpleasant, whole-program transformation into *continuation passing style*) is necessary to implement Church, both to model randomness and to implement `mem` using finite computation. However, this statefulness preserves exchangeability. Understanding the ways in which other stateful language constructs—in particular, primitives for the construction and modification of mutable state—might aid in the description of stochastic processes remains an important area for future work.

2.2 Semantic Correctness

In this section we give formal statements of the claims above, needed to specify the semantics of Church, and sketch their proofs. Let Church$^-$ denote the set of Church expressions that do not include `mem`.

Lemma 2.1. *If $e \in$ Church$^-$ then the weight of* `e` *in a given environment is well-defined and* ≤ 1.

Proof sketch. Arrange the recursive calls to `eval` into a tree with an evaluation at each node and edges connecting successive applications of `eval`—if a node indicates the evaluation of an elementary random procedure there will be several edges descending from this node (one for each possible return value), and these edges are labeled with their probability. A history is a path from root to leaf in this tree and its probability is the product of the labels along the path. Let W_n indicate the sum of probabilities of paths of length n or less. The claim is now that $\lim_{n\to\infty} W_n$ converges and is bounded above by 1. The bound follows because the sum of labels below any random node is 1; convergence then follows from the monotone convergence theorem because the labels are non-negative. □

We next extend the notion of admissibility to arbitrary Church expressions involving `mem`. To compute the probability of an evaluation history we must include the probability of calls to `mem`—that is, the probability of drawing each return value $\mathtt{V}_{val}$. Because there are infinitely many $\mathtt{V}_{val}$, the probability of many histories will then be zero, therefore we pass to equivalence classes of histories. Two histories are *equivalent* if they are the same up to the values bound to $\mathtt{V}_{val}$—in particular they must evaluate all memoized procedures on the same arguments with the same return values. The probability of an equivalence class of histories is the marginal probability over all unused arguments and return values, and this is non-zero. The weight of an expression can now be defined as the sum over equivalence classes of finite histories.

Lemma 2.2. *The admissibility of a Church expression in a given environment is well defined, and any expression* `e` *admissible in environment* `env` *defines a distribution* μ(`e, env`) *on return values of* `(eval 'e env)`.

Proof sketch: The proof is by induction on the number of times `mem` is used. Take as base case expressions without `mem`; by Lemma 2.1 the weight is well defined, so the set of admissible expressions is also well defined.

This function provides persistent class assignments to objects, where classes are symbols drawn from a pool with DP prior:

```
(define drawclass (DPmem 1.0 gensym))
(define class (mem (lambda (obj) (drawclass))))
```

For the beta-binomial model there's a coin weight for each feature/class pair, and each object has features that depend only on it's type:

```
(define coin-weight
  (mem (lambda (feat obj-class) (beta 1 1))) )
(define value
  (mem (lambda (obj feat)
         (flip (coin-weight feat (class obj))) )))
```

For a gaussian-mixture on continuous data (with known variance), we just change the code for generating values:

```
(define mean
  (mem (lambda (obj-class) (normal 0.0 10.0))) )
(define cont-value
  (mem (lambda (obj)
         (normal (mean (class obj)) 1.0) )))
```

The infinite relational model [7] with continuous data is similar, but means depend on classes of two objects:

```
(define irm-mean
  (mem (lambda (obj-class1 obj-class2)
         (normal 0.0 10.0) )))
(define irm-value
  (mem (lambda (obj1 obj2)
         (normal (irm-mean (class obj1) (class obj2))
                 1.0 ))))
```

Figure 3: Church expressions for infinite mixture type models, showing use of the random-world programming style in which objects have persistent properties. Functions `beta` and `normal` generate samples from these standard distributions.

Now, assume $p =$ `(body, args, env)` is an admissible procedure with well defined distribution on return values. The return from `(mem p)` is well defined, because the underlying measure μ(`p, env`) is well defined. It is then straightforward to show that any expression involving `(mem p)`, but no other new memoized procedures, has a well defined weight. The induction step follows.

A subtlety in this argument comes if one wishes to express recursive memoized functions such as:
`(define F (mem (lambda (x) (... F ...))))`.
Prima facie this recursion seems to eliminate the memoization-free base case. However, any recursive definition (or set of definitions) may be re-written without recursion in terms of a fixed-point combinator: `(define F (fix ...))`. With this replacement made we are reduced to the expected situation—application of `fix` may fail to halt, in which case `F` will be inadmissible, but the weight is well defined. □

Lemma 2.2 only applies to expressions involving `mem` for admissible procedures—a relaxation is possible for partially admissible procedures in some situations. From Lemma 2.2 it is straightforward to prove:

Theorem 2.3. *Assume expression* `e` *and procedure* p *are admissible in* `env`*, and let* V *be a random value distributed according to* μ(`e`, `env`)*. If there exists a value* v *in the support of* μ(`e`, `env`) *and* `True` *has non-zero probability under* μ((p v), `env`)*, then the conditional probability*

$$P(V{=}val \mid \texttt{(eval '(}p\ V\texttt{) env)=True})$$

is well defined.

Theorem 2.3 shows that `query` is a well-posed procedure; in Section 4 we turn to the technical challenge of actually implementing query.

3 EXAMPLE PROGRAMS

In this section we describe a number of example programs, stressing the ability of Church to express a range of standard generative models. As our first example, we describe diagnostic causal reasoning in a simple scenario: given that the grass is wet on a given day, did it rain (or did the sprinkler come on)? In outline of this might take the form of the query:

```
(lex-query
 '((grass-is-wet ...)
   (rain ...)
   (sprinkler ...)
 '(rain 'day2)
 '(grass-is-wet 'day2) )
```

where we define a causal model by defining functions that describe whether it rained, whether the sprinkler was on, and whether the grass is wet. The function `grass-is-wet` will depend on both `rain` and `sprinkler`—first we define a noisy-or function:

```
(define (noisy-or a astrength b bstrength baserate)
  (or (and (flip astrength) a)
      (and (flip bstrength) b)
      (flip baserate)))
```

Using this noisy-or function, and a look-up table for various weights, we can fill in the causal model:

```
(lex-query
 '((weight (lambda (ofwhat)
    (case ofwhat
     (('rain-str) 0.9)
     (('rain-prior) 0.3)
      ..etc..)))
   (grass-is-wet (mem (lambda (day)
     (noisy-or
       (rain day) (weight 'rain-str)
       (sprinkler day) (weight 'sprinkler-str)
       (weight 'grass-baserate)))))
```

This deterministic higher-order function defines the basic structure of stochastic transition models:

```
(define (unfold expander symbol)
  (if (terminal? symbol)
      symbol
      (map (lambda (x) (unfold expander x))
           (expander symbol) )))
```

A Church model for a PCFG transitions via a fixed multinomial over expansions for each symbol:

```
(define (PCFG-productions symbol)
  (cond ((eq? symbol 'S)
         (multinomial '((S a) (T a)) '(0.2 0.8)) )
        ((eq? symbol 'T)
         (multinomial '((T b) (a b)) '(0.3 0.7)) ))

(define (sample-pcfg) (unfold PCFG-productions 'S))
```

The HDP-HMM [2] uses memoized symbols for states and memoizes transitions:

```
(define get-symbol (DPmem 1.0 gensym))
(define get-observation-model
  (mem (lambda (symbol) (make-100-sided-die))))
(define ihmm-transition
  (DPmem 1.0 (lambda (state)
               (if (flip) 'stop (get-symbol))
```

```
(define (ihmm-expander symbol)
  (list ((get-observation-model symbol))
        (ihmm-transition symbol) ))
(define (sample-ihmm) (unfold ihmm-expander 'S))
```

The HDP-PCFG [8] is also straightforward:

```
(define terms      '( a  b  c  d))
(define term-probs '(.1 .2 .2 .5))
(define rule-type
  (mem (lambda symbol)
         (if (flip) 'terminal 'binary-production))
(define ipcfg-expander
  (DPmem 1.0
         (lambda (symbol)
           (if (eq? (rule-type symbol) 'terminal)
               (multinomial terms term-probs)
               (list (get-symbol) (get-symbol))))
(define (sample-ipcfg) (unfold ipcfg-expander 'S))
```

Making adapted versions of any of these models [5] only requires stochastically memoizing `unfold`:

```
(define adapted-unfold
  (DPmem 1.0
         (lambda (expander symbol)
           (if (terminal? symbol)
               symbol
               (map (lambda (x)
                      (adapted-unfold expander x))
                    (expander symbol) )))))
```

Figure 4: Some examples of "stochastic transition models".

```
  (rain (mem (lambda (day)
    (flip (weight 'rain-prior)))))
  (sprinkler (mem (lambda (day)
    (flip (weight 'sprinkler-prior))))))
 '(rain 'day2)
 '(grass-is-wet 'day2) )
```

Note that we have used `mem` to make the `grass-is-wet`, `rain`, and `sprinkler` functions persistent. For example, `(= (rain 'day2) (rain 'day2))` is always `True` (it either rained on day two or not), this is necessary since both the query and predicate expressions will evaluate `(rain 'day2)`.

A Bayes net representation of this example would have clearly exposed the dependencies involved (though it would need to be supplemented with descriptions of the form of these dependencies). The Church representation, while more complex, lends itself to intuitive extensions that would be quite difficult in a Bayes net formulation. For instance, what if we don't know the Bernoulli weights, but we do have observations of other days? We can capture this by drawing the weights from a hyper-prior, redefining the weight function to:

```
...(weight (mem (lambda (ofwhat) (beta 1 1))))...
```

If we now query conditioned on observations from other days, we implicitly learn the weight parameters of the model:

```
(lex-query
  '...model definitions...
  '(rain 'day2)
  '(and
    (grass-is-wet 'day1)
    (rain 'day1)
    (not (sprinkler 'day1))
    (grass-is-wet 'day2)) )
```

Going further, perhaps the probability of rain depends on (unknown) types of days (e.g. those with cumulus clouds, cirrus clouds, etc.), and perhaps the probability of the sprinkler activating depends on orthogonal types of days (e.g. Mondays and Fridays versus other days). We can model this scenario by drawing the prior probabilities from two stochastically memoized beta distributions:

```
(lex-query
 '((new-rain-prob
    (DPmem 1.0 (lambda () (beta 1 1))))
   (new-sprinkler-prob
    (DPmem 1.0 (lambda () (beta 1 1))))
   (rain (mem (lambda (day)
    (flip (new-rain-prob)))))
   (sprinkler (mem (lambda (day)
    (flip (new-sprinkler-prob))))))
  ...)
```

With this simple change we have extended the original causal model into an infinite mixture of such models,

in which days are co-clustered into two sets of types, based on their relationship to the wetness of the grass.

In the previous example we left the types of days implicit in the memoizer, using only the probability of rain or sprinkler. In Fig. 3 we have given Church implementations for several infinite mixture models [see 7] using a different idiom—making the types into persistent properties of objects, drawn from an underlying memoized `gensym` (recall that `gensym` is simply a procedure which returns a unique value on each evaluation). Once we have defined the basic structure, `class` to draw latent classes for objects, it is straightforward to define the latent information for each class (e.g. `coin-weight`), and the observation model (e.g. `value`). This basic structure may be used to easily describe more complicated mixture models, such as the continuous-data infinite relational model (IRM) from [7]. Fig. 3 describes forward sampling for these models; to describe a conditional model, these definitions must be made within the scope of a query. For instance, if we wished to query whether two objects have the same class, conditioned on observed features:

```
(lex-query
 '((drawclass (mem 1.0 gensym))
   (class ...)
   (coin-weight ...)
   (value ...))
 '(= (class 'alice) (class 'bob))
 '(and
    (= (value 'alice 'blond) 1)
    (= (value 'bob 'blond) 1)
    (= (value 'jim 'blond) 0)))
```

Another idiom (Fig. 4) allows us to write the common class of "stochastic transition" models, which includes probabilistic context free grammars (PCFGs), hidden Markov models (HMMs), and their "infinite" analogs. Writing the HDP-PCFG [8] and HDP-HMM [2] in Church provides a compact and clear specification to these complicated non-parametric models. If we memoize `unfold` and use this `adapted-unfold` on PCFG transitions we recover the Adaptor Grammar model of [5]; if we similarly "adapt" the HDP-PCFG or HDP-HMM we get interesting new models that have not been considered in the literature.

Fig. 5(top) gives an outline for using Church to represent planning problems. This is based on the translation of planning into inference, given in Toussaint et al. [21], in which rewards are transformed into the probability of getting a single "ultimate reward". Inference on this representation results in decisions which softmaximizes the expected reward. Fig. 5(bottom) fills in this framework for a simple "red-light" game: the state is a light color (red/green) and an integer position, a "go" action advances one position forward except that going on a red light results in being sent back to position 0 with probability `cheat-det`. The goal is to be past position 5 when the game ends; other rewards (e.g. for a staged game) could be added by adding `sp2`, `sp3`, and so on.

```
(define (transition state-action)
  (pair
    (forward-model state-action)
    (action-prior) ))
(define (terminal? symbol) (flip gamma))
(define (reward-pred rewards)
  (flip ((/ (sum rewards) (length rewards)))))
(lex-query
  '((first-action (action-prior))
    (final-state
      (first (unfold transition
        (pair start-state first-action) )))
    (reward-list
      (list (sp1 final-state)
            (sp2 final-state)
            ..etc.. ))
  'first-action
  '(reward-pred reward-list))
```

```
(define (forward-model s-a)
  (pair
    (if (flip 0.5) 'red-light 'green-light)
    (let ((light (first (first s-a)))
          (position (last (first s-a)))
          (action (last s-a)))
      (if (eq? action 'go)
          (if (and (eq? light 'red-light)
                   (flip cheat-det))
              0
              (+ position 1))
          position))))
(define (action-prior) (if (flip 0.5) 'go 'stop))
(define (sp1 state) (if (> (last state) 5) 1 0))
```

Figure 5: Top: The skeleton of planning-as-inference in Church (inspired by [21]). For simplicity, we assume an equal reward amount for each boolean "state property" that is true. Reward is given only when the state reaches a "terminal state", however the stochastic termination decision given by `terminal?` results in an infinite horizon with discount factor `gamma`. Bottom: A specific planning problem for the "red-light" game.

4 CHURCH IMPLEMENTATION

Implementing Church involves two complications beyond the implementation of `eval` as shown in Fig. 1 (which is essentially the same as any lexically scoped, applicative order, pure Lisp [6]). First, we must find a way to implement `mem` without requiring infinite structures (such as the $\mathtt{V}_{val}$). Second, we must implement `query` by devising a means to sample from the appropriate conditional distribution.

To implement `mem` we first note that the countably many $\mathtt{V}_{val}$ are not all needed at once: they can be created as needed, extending the environment `env+`

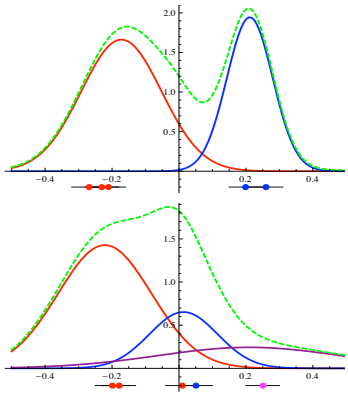


Figure 6: Posterior samples from the infinite gaussian-mixture (with unknown variance) of Section 3, using the collapsed rejection algorithm for `query`. Two datasets are shown (as dots) with mixture components and posterior predictive distribution.

when they are created. (Note that this implementation choices is stateful, but may be implemented easily in full Scheme: the argument/return value pairs can be stored in an association list which grows as need.)[4]

We now turn to `query`. The sampling-based semantics of Church allows us to define a simple rejection sampler from the conditional distribution defining `query`; we may describe this as a Church expression:

```
(define (query exp pred env)
  (let ((val (eval exp env))
    (if (pred val)
        val
        (query exp pred env)))))
```

The ability to write `query` as a Church program—a metacircular [1] implementation—provides a compelling argument for Church's modeling power. However, exact sampling using this algorithm will often be intractable. It is straightforward to implement a collapsed rejection sampler that integrates out randomness in the predicate procedure (accepting or rejecting a `val` with probability equal to the marginal probability that `(p val)` is true). We show results in Fig. 6 of this exact sampler used to query the infinite gaussian-mixture model from Section 3.

In Fig. 7 we show the result of running the collapsed rejection query for planning in the "red-light" game, as shown in Fig. 5 (here `gamma`=0.2, `cheat-det`=0.7). The result is intuitive: when position is near 0 there is little to lose by "cheating", as position nears 5 (the goal line) there is more to loose, hence the probability of cheating decreases; once past the goal line there is nothing to be gained by going, so the probability of cheating drops sharply. Note that the "soft-max" formulation of planning used here results in fairly random behavior even in extreme positions.

[4]A further optimization implements `DPmem` via the Chinese restaurant process representation of the DP [15].

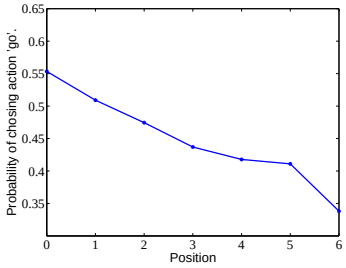


Figure 7: Results from planning in the "red-light" game (Fig. 5), showing the probability of "cheating" (going when the light is red) versus position. The goal is to end the game past position 5.

4.1 A Metropolis-Hastings Algorithm

We now present a Markov chain Monte Carlo algorithm for approximately implementing `query`, as we expect (even collapsed) rejection sampling to be intractable in general. Our algorithm executes stochastic local search over evaluation histories, making small changes by proposing changes to the return values of elementary random procedures. These changes are constrained to produce the conditioned result, collapsing out the predicate expression via its marginal probability[5]. The use of evaluation histories, rather than values alone, can be viewed as an extreme form of data-augmentation: all random choices that lead to a value are made explicit in its history.

The key abstraction we use for MCMC is the *computation trace*. A computation trace is a directed, acyclic graph composed of two connected trees. The first is a tree of evaluations, where an evaluation node points to evaluation nodes for its recursive calls to `eval`. The second is a tree of environment extensions, where the node for an extended environment points to the node of the environment it extends. The evaluation node for each `(eval 'e env)` points to the environment node for `env`, and evaluation nodes producing values to be bound are pointed to by the environment extension of the binding. Traces are in one-to-one correspondence with equivalence classes of evaluation histories, described earlier[6]. Fig. 8 shows the fragment of a computation trace for evaluation of the expression `((lambda (x) (+ x 3)) (flip))`.

For each elementary random procedure `p` we need a Markov chain transition kernel $K_{\texttt{p}}$ that proposes a new return value for that procedure given its current arguments. A generic such kernel comes from re-

[5]Handling the rejection problem on chain initialization (and queries across deterministic programs, more generally) is a challenge. Replacing all language primitives (including `if`) with noisy alternatives and using tempering techniques provides one general solution, to be explored in future work.

[6]Also note that the acyclicity of traces is a direct result of the purity of the Church language: if a symbol's value were mutated, its environment would point to the evaluation node that determined its new value, but that node would have been evaluated in the same environment.

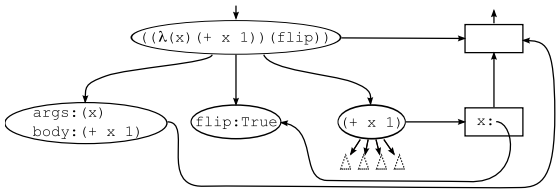


Figure 8: A schematic computation trace.

evaluating `(eval '(`p `args) env)`; however, a proper Church standard library could frequently supply more efficient proposal kernels for particular procedures (for instance a drift kernel for `normal`). Our requirement is that we are able to sample a proposal from $K_{\mathtt{p}}$ as well as evaluate its transition probability $q_{\mathtt{p}}(\cdot|\cdot)$.

If we simply apply $K_{\mathtt{p}}$ to a trace, the trace can become "inconsistent"—no longer representing a valid evaluation history from `eval`. To construct a complete Metropolis-Hastings proposal from $K_{\mathtt{p}}$, we must keep the computation trace consistent, and modify the proposal probabilities accordingly, by recursing along the trace updating values and potentially triggering new evaluations. For example, if we change the value of `flip` in `(if (flip)` e_1 e_2`)` from `False` to `True` we must: absorb the probability of `(eval` e_2 `env)` in the reverse proposal probability, evaluate e_1 and attach it to the trace, and include the probability of the resulting sub-trace in the forward proposal probability. (For a particular trace, the probability of the sub-trace for expression e is the probability of the equivalence class of evaluation histories corresponding to this sub-trace.) The recursions for trace consistency and proposal computation are delicate but straightforward, and we omit the details due to space constraints[7]. Each step of our MCMC algorithm[8] consists of applying a kernel $K_{\mathtt{p}}$ to the evaluations of a randomly chosen elementary random primitive in the trace, updating the trace to maintain consistency (collecting appropriate corrections to the proposal probability), and applying the Metropolis-Hastings criterion to accept or reject this proposal. (This algorithm ignores some details needed for queries containing nested queries, though we believe these to be straightforward.)

We have implemented and verified this algorithm on several examples that exercise all of the recursion and update logic of the system. In Fig. 9 we have shown convergence results for this algorithm running on the simple "sprinkler" example of Section 3.

[7]We implemented our MCMC algorithm atop the Blaise system [3], which simplifies these recursively triggered kernel compositions.

[8]At the time of writing we have not implemented this algorithm for programs that use `mem`, though we believe the necessary additions to be straightforward.

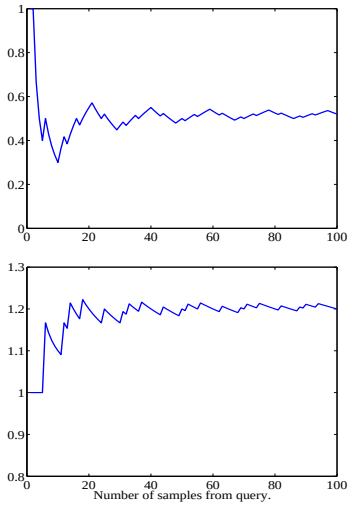


Figure 9: Convergence of one run of the MCMC algorithm on the "sprinkler" example. (Each sample from query uses 30 MCMC steps.) Top: The probability of `rain`. Bottom: The expected value of `(+ (rain) (sprinkler))`, showing explaining away. The sum is slightly above 1.0 because one cause is usually present, but both rarely are.

5 DISCUSSION

While Church builds on many other attempts to marry probability theory with computation, it is distinct in several important ways. First, Church is founded on the lambda calculus, allowing it to represent higher-order logic and separating it from many related languages. For example, unlike several widely used languages grounded in propositional logic (e.g. BUGS [9]) and first-order logic (e.g. the logic programming approaches of [13, 19], BLOG [12], and Markov logic [18]), generative processes in Church are first-class objects that can be arbitrarily composed and abstracted. The example programs in Section 3 illustrate the representational flexibility of Church; while some of these programs may be naturally represented in one or another existing language, we believe that no other language can easily represent all of these examples.

The stochastic functional language IBAL [14], based on the functional language ML, is quite similar to Church, but the two languages emphasize different aspects of functional programming. Other related work includes non-determistic [11] and weighted non-deterministic [16] extensions to Lisp. Unlike these approaches, the semantics of Church is fundamentally sampling-based: the denotation of admissible expressions as distributions follows from the semantics of evaluation rather than defining it. This semantics, combined with dynamic typing (cf. static typing of ML), permits the definition and exact implementation of `query` as an ordinary Church procedure, rather than a special transformation applied to the distribution denoted by a program. Because `query` is defined via sampling, describing approximate inference is particularly natural within Church.

A number of the more unusual features of Church as a

stochastic programming language derive from its basis in Lisp. Since `query` and `eval` are the basic constructs defining the meaning of Church expressions, we have a metacircular [17] description of Church within Church. This provides clarity in reasoning about the language, and allows self-reflection within programs: queries may be nested within queries, and programs may reason about programs. Church expressions can serve both as a declarative notation for uncertain beliefs (via the distributions they represent) and as a procedural notation for stochastic and deterministic processes (via evaluation). Because expressions are themselves values, this generalizes the Lisp unification of programs and data to a unification of stochastic processes, Church expressions, and uncertain beliefs. These observations suggest exciting new modeling paradigms. For instance, `eval` nested within `query` may be used to learn programs, where the prior on programs is represented by another Church program. Issues of programming style then become issues of description length and inductive bias. As another example, `query` nested within `query` may be used to represent an agent reasoning about another agent.

Of course, Church's representational flexibility comes at the cost of substantially increased inference complexity. Providing efficient implementations of `query` is a critical challenge as our current implementation is not yet efficient enough for typical machine learning applications; this may be greatly aided by building on techniques used for inference in other probabilistic languages [e.g. 10, 14, 12]. For example, in Church, exact inference by enumeration could be seen as a program analysis that transforms expressions involving `query` into expressions involving only `eval`; identifying and exploiting opportunities for such transformations seems appealing.

Probabilistic models and stochastic algorithms are finding increasingly widespread use throughout artificial intelligence and cognitive science, central to areas as diverse as vision, planning, and natural language understanding. As their usage grows and becomes more intricate, so does the need for formal languages supporting model exchange, reuse, and machine execution. We hope Church represents a significant step toward this goal.

Acknowledgements

The authors would like to thank Gerry Sussman, Hal Abelson, Tom Knight, Brian Milch, David McAllester and Alexey Radul for helpful discussions. This work was funded in part by a grant from NTT Communication Sciences Laboratory.

References

[1] H. Abelson and G. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, 1996.

[2] M.J. Beal, Z. Ghahramani, and C.E. Rasmussen. The infinite hidden Markov model. *NIPS 14*, 2002.

[3] K. A. Bonawitz. *Composable Probabilistic Inference with* Blaise. PhD thesis, MIT, 2008.

[4] A. Church. A Set of Postulates for the Foundation of Logic. *The Annals of Mathematics*, 33(2):346–366, 1932.

[5] M. Johnson, T. Griffiths, and S. Goldwater. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. *NIPS 19*, 2007.

[6] R. Kelsey, W. Clinger, and J. Rees (eds.). Revised^5 Report on the Algorithmic Language Scheme. *Higher-Order and Symbolic Computation*, 11(1):7–105, 1998.

[7] C. Kemp, J.B. Tenenbaum, T.L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. *Proc. 21st Natl Conf. Artif. Intell., AAAI Press*, 2006.

[8] P. Liang, S. Petrov, M.I. Jordan, and D. Klein. The Infinite PCFG using Hierarchical Dirichlet Processes. *Proc. EMNLP-CoNLL*, 2007.

[9] D.J. Lunn, A. Thomas, N. Best, and D. Spiegelhalter. WinBUGS-A Bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing*, 10(4):325–337, 2000.

[10] D. McAllester, B. Milch, and N. D. Goodman. Random-world semantics and syntactic independence for expressive languages. Technical Report MIT-CSAIL-TR-2008-025, Massachusetts Institute of Technology, 2008.

[11] J. McCarthy. A Basis for a Mathematical Theory of Computation. In *Computer Programming and Formal Systems*, pages 33–70, 1963.

[12] B. Milch, B. Marthi, S. Russell, D. Sontag, D.L. Ong, and A. Kolobov. BLOG: Probabilistic models with unknown objects. *Proc. IJCAI*, 2005.

[13] S. Muggleton. Stochastic logic programs. In L. de Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.

[14] A. Pfeffer. IBAL: A probabilistic rational programming language. *Proc. IJCAI*, 2001.

[15] J. Pitman. Combinatorial stochastic processes, 2002. Notes for Saint Flour Summer School.

[16] A. Radul. Report on the probabilistic language scheme. Technical Report MIT-CSAIL-TR-2007-059, Massachusetts Institute of Technology, 2007.

[17] J.C. Reynolds. Definitional interpreters for higher-order programming. *ACM Annual Conference*, pages 717–740, 1972.

[18] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1):107–136, 2006.

[19] T. Sato and Y. Kameya. PRISM: A symbolic-statistical modeling language. In *International Joint Conference on Artificial Intelligence*, 1997.

[20] J. Sethuraman. A Constructive definition of Dirichlet priors. *Statistica Sinica*, 4, 1994.

[21] M. Toussaint, S. Harmeling, and A. Storkey. Probabilistic inference for solving (PO)MDPs. Technical Report EDI-INF-RR-0934, University of Edinburgh, 2006.

Latent Topic Models for Hypertext

Amit Gruber
School of CS and Eng.
The Hebrew University
Jerusalem 91904 Israel
amitg@cs.huji.ac.il

Michal Rosen-Zvi
IBM Research Laboratory in Haifa
Haifa University, Mount Carmel
Haifa 31905 Israel
rosen@il.ibm.com

Yair Weiss
School of CS and Eng.
The Hebrew University
Jerusalem 91904 Israel
yweiss@cs.huji.ac.il

Abstract

Latent topic models have been successfully applied as an unsupervised topic discovery technique in large document collections. With the proliferation of hypertext document collection such as the Internet, there has also been great interest in extending these approaches to hypertext [6, 9]. These approaches typically model links in an analogous fashion to how they model words - the document-link co-occurrence matrix is modeled in the same way that the document-word co-occurrence matrix is modeled in standard topic models.

In this paper we present a probabilistic generative model for hypertext document collections that explicitly models the generation of links. Specifically, links from a word w to a document d depend directly on how frequent the topic of w is in d, in addition to the in-degree of d. We show how to perform EM learning on this model efficiently. By not modeling links as analogous to words, we end up using far fewer free parameters and obtain better link prediction results.

1 Introduction

The need to automatically infer the different topics discussed in a corpus arises in many applications ranging from search engines to summarization software. A prominent approach is modeling the corpus with a latent topic model where each document is viewed as a mixture of latent topics or factors, and the factors, shared by the whole corpus, are related to the terms or words appearing in the documents.

Many of the topic models share the "bag of words" assumption where each document is represented as a histogram of terms, ignoring the order of terms and the internal structure of the documents. The entire corpus is represented as a document-term co-occurrence matrix. Semantic analysis is done by projecting the document-term co-occurrence matrix onto a lower dimensional factor space. In algebraic methods such as Latent Semantic Analysis [7] it is projected onto a linear factor space using SVD. In statistical methods such as Probabilistic LSA [13], Latent Dirichlet Allocation [3] or the somewhat more general formalism, Discrete PCA [4] the document-term co-occurrence matrix is projected onto a simplex by maximizing the observations likelihood. In recent years these latent topic models have been extended in various ways. In particular, correlation between topics [2] and their dynamics over time [1] have been directly modeled. The use of additional information provided in the corpus such as authorship information has been studied [18]. In addition, novel models that depart from the bag of words assumption and do consider the internal ordering of the words in sentences within a document have been developed. These models combine local dependencies in various ways; for example, combining n-grams with a hierarchical topic model [19], modeling syntax [10] and modeling the continuous drift from one topic to another within a document [12]

In this paper, we address the question of how to enrich the model by considering links between documents, such as hyperlinks in hypertext or citations in scientific papers. With the emergence and rapid growth of the World Wide Web, hypertext documents containing links to other documents have become ubiquitous. The connectivity between documents has proven to play an important role in determining the importance and relevance of a document for information retrieval or the interest of a certain user in it [17, 5, 14]. In particular, Dietz at al. [8] have recently proposed a generative topic model for the prediction of citation influences, called the citation influence model. It models the particular structure of paper citations where the citations graph can be described by a directed acyclic graph

(DAG); a setting that does not hold in the case of the World Wide Web and other hypertext corpora.

There are few previous works that extend topic models to include link information. Cohn and Hofmann [6] introduce a joint probabilistic model for content and connectivity. The model is based on the assumption that similar decomposition of the document term co-occurrence matrix can be applied to the cite-document co-occurrence matrix in which each entry is a count of appearances of a linked-document (or citation) in a source document. In this approach, links are viewed as additional observations and are analogous to additional words in the vocabulary, but with different weight when estimating the topic mixture of the document. Erosheva et al. [9] also makes use of a decomposition of term-document and citation-document co-occurrence matrices by extending the LDA model to include a generative step for citations. Note that these models only learn from the co-occurrence matrix of citations without exploiting the information conveyed by the cited documents text. Thus, if the citation-document co-occurrences matrix is very sparse, the generalization power of the models is very limited.

In this paper, we suggest a novel generative model for hypertext document collection that we name the latent topic hypertext model (LTHM). Our approach includes direct modeling of real-world complex hypertext collections in which links from every document to every document may exist, including a self-reference (a document linking to itself). We model a link as an entity originating from a specific word (or collection of words) and pointing to a certain document. The probability to generate a link from a source document d to a target document d' depends on the topic of the word from which the link is originating, on the importance of the target document d' (estimated roughly by the in-degree) and on the topic mixture of the target document d'. In this way, an observed link directly affects the topic mixture estimation in the target document as well as the source document. Moreover, the non-existence of a link between two documents is an observation that serves as evidence for the difference between the topic mixtures of the documents.

We introduce the LTHM and related models in Section 2 and describe the approximate inference algorithm in Section 3. Experimental results obtained by learning two datasets are provided in Section 4. Finally, we discuss the results in Section 5.

2 The latent topic hypertext model

The topology of the World Wide Web is complicated and unknown. The corpus we work with is a subset of the World Wide Web and its topology can be arbitrary accordingly. By no means can we assume it forms a DAG. Therefore, we would like to allow each document to link to any other document, allowing for loops, i.e. directed cycles of links originating in a certain document and ending in the same document. In particular, we would like to allow for self loops with links where a document links to itself. The solution is a generative model that consists of two stages. In the first stage, the document content (the words) is created. After the text of all the documents has been created, the second stage of creating links takes place.

The contribution of this paper is in modeling link generation and suggesting an approximate inference algorithm for studying it. The text in the documents can be generated using several of the various models mentioned in section 1. For simplicity, we describe text generation (and inference, accordingly) using LDA [3]. In the following section, we first briefly review the LDA model (2.1). Second, we describe the second stage of link generation (2.2). Finally, we discuss related models (in section 2.3).

2.1 Document generation (LDA)

According to the LDA model, a collection of documents is generated from a set of K latent factors or topics. One of the main assumptions in the model is that for each topic there is a single multinomial random variable β that defines the probability for a word given a topic for all documents in the collection. Each document is characterized by a particular mixture of topic distribution defined by the random variable θ. The generation of the N_d words of each document d in a corpus contains two stages: first, a hidden topic z is selected from a multinomial distribution defined by θ. Second, given the topic z, a word w is drawn from the multinomial distribution with parameters β_z. Figure 2.1a illustrates the generative model.

Formally, the model can be described as:

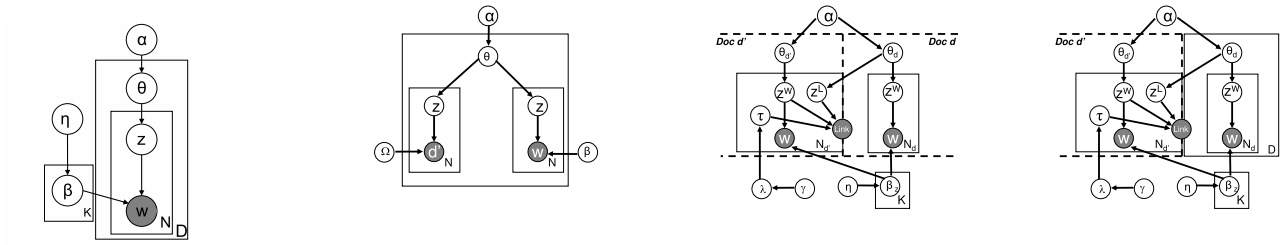
1. For each topic $z = 1, ..., K$ choose W dimensional $\beta_z \sim \text{Dirichlet}(\eta)$
2. For each document $d = 1, ..., D$

   Choose K dimensional $\theta \sim \text{Dirichlet}(\alpha)$

   For each word w_i, indexed by $i = 1, ..N_d$

   Choose a topic $z_i^W \sim \text{Multinomial}(\theta_d)$

   Choose a word $w_i \sim \text{Multinomial}(\beta_{z_i^W})$



(a) (b) (c) (d)

Figure 1: **a.** The LDA model. **b** The link-LDA model. **c.** The LTHM model in a scenario of generating links from document d' to document d. **d.** The LTHM model in a scenario of generating links from document d' to any other document in the collection of D documents.

2.2 Link generation

We assume that links originate from a word, and each word can have at most one link associated with it[1]. For simplicity, we restrict the discussion to the case where a link is anchored to a single word. The generalization to the case where the link is anchored to a sequence of words can be carried out by forcing the topics of these words to be identical, as proposed in [12]. The generation of links is carried out by iterating over all the words in the document and for each word determining whether to create a link and if so, what is the target document.

Let us limit the discussion first to the case where the corpus contains two documents, d and d', and links are generated from words in document d' to document d. When iterating over the words in d', at the i^{th} word, we need to decide whether to create a link from w_i to d or not. This decision contains two steps (at most, as sometimes the first step is sufficient to determine that no link needs to be created). The first step is drawing at random a variable τ_i from a multinomial λ. In general, τ_i can take values from 0 to D, and in this degenerated example it can take two values: 0 indicates no link and d indicates a link to document d. Only if $\tau_i = d$ do we consider adding a link to document d and then proceed to the next step, which is randomly drawing the topic of the link, z^L. The topic assignment z^L is drawn from θ_d, the mixture of topics of the document d. A link is created iff $z_i^W = z^L$, Figure 2.1c illustrates the full generative model for this degenerated example.

The generalization to the (still degenerate) case of generating links from a single document d' to any other document in a collection of D documents is illustrated in Figure 2.1d. In this case, the generation of links from words in document d' starts by selecting $\tau_i \in \{1 \dots D, \emptyset\}$ for every word $i = 1..N_{d'}$ of the document d'. τ is drawn at random from $\lambda \in R^{D+1}$, a multinomial distribution indicating the probability of considering a link to each one of the D documents or not having a link at all. It is a measure of the importance (or in-degree) of the documents in the corpus. λ itself is drawn from the hyperparameter γ. The Dirichlet prior $\gamma \in R^{D+1}$ is not symmetric and favors not creating links, as most words do not have an associated link: $\gamma_i \ll \gamma_\emptyset$ for $i = 1 \dots D$. Also, note that links from a document to itself are allowed in this model (as well as in real life).

The most general case, in which every document can contain words linked to any other document, is generated by sequentially going through all words and all documents and drawing at random the corresponding τs and z^Ls in the way described above.

Formally, the generative process is:

1. Choose $D+1$ dimensional $\lambda \sim \text{Dirichlet}(\gamma)$
2. For each document $d = 1, ..., D$

   For each word w_i, indexed by $i = 1, ..N_d$

   Choose $\tau_i \in \{1 \dots D, \emptyset\} \sim \text{Multinomial}(\lambda)$

   If $\tau_i \neq \emptyset$ choose a topic $z_L \sim \text{Multinomial}(\theta_{\tau_i})$

   If $z^L = z_i^W$ create a link $L_i = \tau_i$ from

   word i to document τ_i

2.3 Related Models

Both models of [6] and [9] – that we refer to as link-PLSA and link-LDA, respectively, following [16]'s suggestion – are make of the citation-document co-occurrence matrix in a similar manner. We focus on

[1] If a link is anchored to an image, for example, we could substitute a fictitious word for that link.

the link-LDA model that is somewhat closer to our model. According to this approach two types of observed variables are modeled: words in documents and citation in documents. The generation of these variables is carried out by first selecting a mixture of topics for each of the documents and then for each of the words and citations on the document generating a hidden topic from which the observation is selected at random from the β_z in the case of words and from Ω_z in the case of citations; Here $z = 1, ..., K$. The model is illustrated in Figure 2.1b.

Formally, the model can be described as:

For each document $d = 1, ..., D$

Choose K dimensional $\theta \sim \text{Dirichlet}(\alpha)$

For each word w_i, indexed by $i = 1, ..N_d$

Choose a topic $z_i \sim \text{Multinomial}(\theta_d)$

Choose a word $w_i \sim \text{Multinomial}(\beta_{z_i})$

For each citation d'_i, indexed by $i = 1, ..L_d$

Choose a topic $z_i \sim \text{Multinomial}(\theta_d)$

Choose a citation $d'_i \sim \text{Multinomial}(\Omega_{z_i})$

Note that in the LTHM, the probability to create a link given topic is $\Pr(link = d|z) = \lambda_d \theta_d(z^W)$. There are only D additional parameters $\lambda_1 \dots \lambda_D$ to denote the document importance for link creation, whereas in the link-LDA model there are DK additional parameters, $\Omega_{d,z} = \Pr(d' = d|z)$. Also, in LTHM, the very existence or non-existence of a link is an observation, while this is not explicitly modeled by the link-LDA. Moreover, according to the LTHM, a link shares the same topic with the word it originates from and at the same time affects the topic mixture in the cited document.

3 Approximate Inference

Exact inference in hierarchical models such as LDA and PLSA is intractable due to the coupling of the latent topics and the mixing vectors β, θ. The hypertext model presented in this paper shares this coupling and adds a unique coupling between topic mixing vectors; hence, exact inference is intractable in it as well. In recent years, several alternatives for approximate inference in such models have been suggested: EM [13] or variational EM [3], Expectation propagation (EP) [15] and Monte-Carlo sampling [18, 11]. Unlike other hypertext topic models, in LTHM not only the identities of the ends of a link are observations, but also the link's very existence (or non-existence). Taking into account the non-existence of links in sampling-based inference necessitates further approximations. We therefore perform inference using EM.

EM deviates from fully Bayesian methods by distinguishing between latent variables and parameters of the model. The latent variables are the latent topic of a word, z^W, the latent topic involved in link generation, z^L, and the variable τ. The parameters of the model are the topic mixing vectors θ_d, the word mixing vectors β_z and the document link importance parameter λ_d. The Dirichlet hyperparameters α, η and γ are fixed.

In the link generation process, unless a link is created, the value of τ is unknown. It might have not been created because $\tau = \emptyset$ or because of topic mismatch between the source document and any other document. For this reason, we need to consider all possible options with their probability during inference: for each source document d and each word in it from which there is no outgoing link, we need to consider all D possible z^L variables. The number of the potential latent variables z^L is $D \sum_d Nd$ which is quadratic in the number of documents. It is therefore infeasible to compute explicitly the posterior distribution of each one of these latent variables. However, in the M-step, only aggregations of these posterior distributions are needed. The required aggregations can be computed efficiently (in time linear in the size of the corpus) by taking advantage of symmetries in the model as described in section 3.2 and in the appendix. We begin with the M-step equations, detailing what are the required expectations. Then we describe how the required posteriors and aggregations are computed in the E-step.

3.1 M-step

In the M-step, MAP estimators for the parameters of the model, θ_d, β_z and λ are found. Let $G_{z,w}$ denote the number of occurrences of a word w with topic $z^W = z$. The update rule for $\beta_{z,w}$ is identical to that in standard LDA:

$$\beta_{z,w} \quad \propto \quad E(G_{z,w}) + \eta_w - 1 \tag{1}$$

The MAP estimator for θ_d takes into account topics of words and links that were drawn from θ_d. The word topics, z^W, are drawn from θ_d for each of the words in document d. The link topics are the topics $z^L_{d',i,d}$ drawn from θ_d when considering a link from any other document d' to d. These are the cases where $\tau_{d',i} = d$ for any d', i regardless of whether the link has been created or not. For the purpose of inference, we count

the topics $z^L_{d',i,d}$ separately for links and for non-links. Let $F_{d,z}$ denote the number of occurrences of a topic z associated with any word in document d. Let $V_{d,z}$ be the number of occurrences of a topic z associated with any incoming link of document d. Let $U_{d,z}$ be the number of times $\tau_{d',i} = d$ but the topic generated for the link by document d, $z^L_{d',i,d}$, does not match the topic of the ith word in the the document d', $z^W_{d',i}$ and therefore a link has not been created.

$$\theta_{d,z} \propto E(F_{d,z}) + E(V_{d,z}) + E(U_{d,z}) + \alpha_z - 1 \quad (2)$$

Note that in the standard LDA model, we would have just the first term (the expected number of times topic z appears in document d) and the Dirichlet prior. In the LTHM, we add two more terms which model the influence of links (or non-links) on the topic distribution.

The computation of $E(V_{d,z})$ and $E(U_{d,z})$ is described in section 3.2.

The MAP estimator for λ is

$$\lambda_d \propto E(T_d) + \gamma_d - 1 \quad (3)$$

$$\lambda_\emptyset \propto \sum_d N_d - \sum_d E(T_d) + \gamma_\emptyset - 1 \quad (4)$$

Where T_d is the number of times that $\tau_{d',i} = d$ for any d' and any word i in it (this includes the case of $d' = d$ where a self link is considered). Notice that $T_d = \sum_z (V_{d,z} + U_{d,z})$. The normalization factor in equations 3 and 4 includes the term $\lambda_\emptyset$, the most frequent case that there is no link at all.

3.2 E-step

In the E-step, expectations required for the M-step are computed with respect to the posterior distribution of the latent variables. The expectations required for the M-step are $E(G_{d,z}), E(F_{z,w}), E(V_{d,z}), E(U_{d,z})$ and $E(T_d)$.

$E(G_{d,z})$ is the expected number of occurrences of a topic z in document d as a topic of word and $E(F_{z,w})$ is the expected number of occurrences of a word w with topic z:

$$E(G_{d,z}) = \sum_{i=1}^{N_d} \Pr(z^W_{d,i} = z|\bar{w}, \bar{L}) \quad (5)$$

$$E(F_{k,z}) = \sum_{d=1}^{D}\sum_{i=1}^{N_d} \Pr(z^W_{d,i} = z, w_{d,i} = w|\bar{w}, \bar{L}) \quad (6)$$

where $\bar{w} = w_1 \dots w_{N_d}$ and $\bar{L} = L_1 \dots L_{N_d}$. The posterior distribution of z^W is explicitly computed, taking into account words and links (or the non-existence of a link) as observations.

$$\Pr(z^W_{d,i} = z|\bar{w}, \bar{L}) \propto \theta_d(z) \Pr(L_{d,i}|z^W_{d,i} = z)\phi_z(w_{d,i}) \quad (7)$$

where $\Pr(L_{d,i}|z^W_{d,i} = z)$, the probability of a link observation is

$$\Pr(link(d,i) \to d'|z^W_{d,i} = z; P) = \lambda_{d'}\theta_{d'}(z) \propto \theta_{d'}(z)$$

if a there is a link from word i in document d to document d', and

$$\Pr(no - link(d,i)|z^W_{d,i} = z; P) = 1 - \sum_{d'} \lambda_{d'}\theta_{d'}(z)$$

if there is no link associated with word i in document d.

Naïve computation of $E(V_{d,z})$ and $(E(U_{d,z})$ would require estimating the posterior distributions of $\tau_{d',i}$ and $z^L_{d',i,d}$ for all triplets (d', i, d). As mentioned before, explicit computation of these posteriors is infeasible due to large number of these variables. Rather than computing this posterior distribution explicitly over z^L and τ, only the aggregations $E(V_{d,z}), E(U_{d,z})$ are computed.

$E(V_{d,z})$ is the expected number of occurrences of links incoming to document d with topic z. In the case where a link exists, the posterior distributions of z^L and the corresponding z^W are equal; hence, $V_{d,z}$ can be computed by summing posterior probabilities of z^W:

$$E(V_{d,z}) = \sum_{(d',i)\in A_d} \Pr(z^L_{d',i,d} = z|O, P) = \sum_{(d',i)\in A_d} \Pr(z^W_{d',i} = z|O, P) \quad (8)$$

where $A_d = \{(d', i) : link(d', i) \to d\}$, O is the set of all observations and P is the model parameters.

$E(U_{d,z})$ is the expected number of times $\tau_{d',i} = d$ for any d', i in the corpus, but $z^L_{d',i,d} \neq z^W_{d',i}$. The basic idea in the computation of $E(U_{d,z})$ is that it factors into topic dependent terms and document-topic dependent terms. The topic dependent terms can be computed in a single linear pass over the corpus (in each iteration). The document-topic dependent terms are specific to each $U_{d,z}$. Combining them with the topic dependent terms to compute $U_{d,z}$ is done in a constant number of operations (for each d, z). The computation is detailed in the appendix.

Finally, after $E(V_{d,z})$ and $E(U_{d,z})$ have been computed,

$$E(T_d) = \sum_z [E(V_{d,z}) + E(U_{d,z})] \quad (9)$$

Despite the quadratic number of latent variables, the runtime of both E and M steps is linear in the size of the corpus times the number of topics.

There are a number of extensions to the LDA model that can be considered here, as the approximate inference algorithm described above can be easily adapted for many of the alternatives mentioned in section 1. For example, suppose one wishes to model the text with the HTMM [12], the difference would be in the computation of the posterior of word topics, $\Pr(z^W|w_1 \dots w_{N_d}, L_1 \dots L_{N_d})$. In HTMM, this posterior would be computed using the forward-backward algorithm, considering both words and links as the topic emission probabilities. Alternatively, if one wishes to make use of authorship information by applying the Author-Topic model [18], it would require to consider an additional latent variable x for the authorship of each word and compute posterior probabilities $\Pr(x, z^W|w_1 \dots w_{N_d}, L_1 \dots L_{N_d})$ and modify the definition of θ_d. Yet, the modeling of links stays very similar; in addition to latent topic of the link, z^L, only a latent author to the link needs to be selected, x_L.

4 Experiments

In this section, we explore the relations between links and topics discovered by LTHM and evaluate its predictive power with respect to links. We compare LTHM's link prediction with previous approaches for combining links: link-PLSA[6] and link-LDA[9]. We also compare to link prediction by a non-topic method, based only on the frequency a web page is linked, ignoring the contents of the text in the corpus. For this comparison, we use two datasets of web pages: the webkb dataset (8282 documents with 12911 links) and a small dataset of Wikipedia web pages (105 documents with 799 links).

We begin with an example of the strong relationships between topics and links in the Wikipedia dataset learned by LTHM. The Wikipedia dataset is a collection of 105 web pages with 799 links between the pages in the dataset. We downloaded these web pages from Wikipedia by crawling within the Wikipedia domain, starting from the NIPS[2] Wikipedia page. We have made the data set available online at: http://www.cs.huji.ac.il/∼amitg/lthm.html. We used a vocabulary of 2247 words and trained LTHM with 20 hidden aspects. Figure 4 shows four of the hidden topics found by the model in the Wikipedia dataset. For each topic we show the ten most probable words and two most probable links. Topic 1 discusses neural networks, and the two most related links to it (links with high probability to be generated from the topic). Similarly, topic 2 is about speech and pattern recognition. Topic 3 is about cities (Denver and Vancouver, the current and previous venues of the nips conference). topic 4 is about cognitive science and neuroscience. All these topics have related links. Due to lack of space, we show only four example topics, but all 20 topics have clear interpretation and relevant suggested links. A complete list of the topics with top words and top links can be found at http://www.cs.huji.ac.il/∼amitg/lthm.html.

We found that the LDA topics on this dataset were of comparable quality, but the assignment of topics to documents can be different in LDA and LTHM, especially for short documents. Table 1 shows a comparison of the document topic vector θ_d for two short Wikipedia documents, "Journal of Machine Learning" and "Random Forests", in LDA and LTHM. All topics with $\theta_d > 0.05$ are shown. Since LTHM uses the link information, it assigns more weight to the relevant topics.

For quantitative evaluation, we compare LTHM vs. the topic models link-PLSA [6] and link-LDA [9] and a frequency-based method in the task of link prediction. The frequency-based method ranks the documents in the corpus according to the number of time they were linked to from other documents. This ranking serves as the link prediction for all the documents. This prediction is the same for all the documents in the corpus and does not depend on the topics of the source document.

For these experiments we use the Wikipedia dataset and the webkb dataset. The webkb dataset (available online at: http://www.cs.cmu.edu/∼webkb) consists of 8282 html pages. For this dataset, we used a dictionary of 2304 words (built according to their frequency in the data and removing stop words). We extracted 12911 links where both ends of the links belong to the webkb corpus. We split each data set into a train set consisting of 90% of the documents and a test set of the remaining 10%. During training, the text of all the documents is provided to all the algorithms, but only the links originating from the documents in the train set are visible during training. Both dataset are learned with 20 hidden aspects. During test, for each test document d we sort all documents d' in the corpus according to the probability of having a link from d (outgoing from any word in d) to d'.

Figures 3 and 5 show several measures of the performance of the different algorithms. The first measure is the percentage of documents in the test set for which at least one link prediction among the top N is a true link. The motivation for this measure is the following question: Suppose we want to suggest to an author of

[2] At the time being, there is no UAI Wikipedia page.

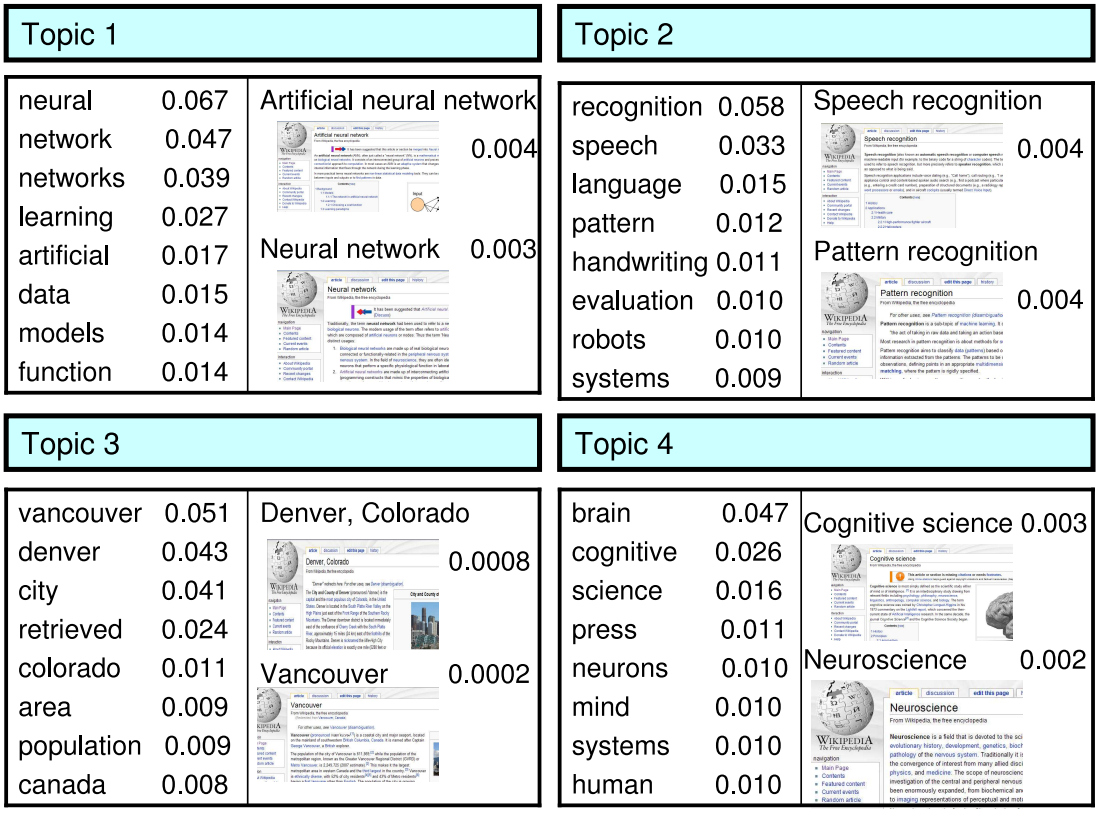


Figure 2: Four example topics learned by LTHM and the links related to them. For each topic, the ten most probable words are shown along with the two links most probable to be generated from that topic. Next to each word and each link is its probability to be generated from the given topic.

Journal Of Machine Learning Research.html

LDA

topic prob	top words
0.1504	”search”,”article”,”navigation”
0.0798	”press”,”university”,”new”
0.0652	”learning”,”machine”,”algorithms”
0.0594	”fixes”,”skins”,”import”
0.0533	”model”,”regression”,”reasoning”

LTHM

topic prob	top words
0.4136	”learning”,”machine”,”engineering”
0.0943	”card”,”conference”,”credit”

Random Forests.html

LDA

topic prob	top words
0.1416	”probability”,”distribution”,”variables”
0.1194	”data”,”mining”,”predictive”
0.0757	”learning”,”machine”,”algorithms”
0.0542	fixes”,”skins”,”import”
0.0527	stock”,”market”,”price”
0.0527	search”,”article”,”navigation”

LTHM

topic prob	top words
0.2076	”linear”,”function”,”training”
0.1921	”fuzzy”,”regression”, ”model”
0.1178	”bayesian”, ”model”, ”network”
0.0547	”carlo”,”monte”,”genetic”
0.0524	”learning”,”machine”,”engineering”

Table 1: A comparison of the document topic vector θ_d for two short Wikipedia documents “Journal of Machine Learning” and “Random Forests” in LDA and LTHM. All topics with $\theta_d > 0.05$ are shown. Since LTHM uses the link information, it assigns more weight to the relevant topics.

a web page other documents to link to. If we show this author N suggestions for links, will s/he use at least one of them? The other measures we use are precision (Among the top N predictions, what is the percentage of true links?) and recall (What percentage of the true links are included in the top N predictions?).

Figure 3 shows that LTHM outperforms all three other methods with respect to all three performance measures. Both link-PLSA and link-LDA do worse than the frequency-based method. This result may seem surprising at first, as these methods are more general than the frequency-based method. In particular, they could fit the relative frequency of each document as its probability to be drawn from any topic. In this case, they would predict the same as the frequency-based method. When we inspect the performance of these methods on the train set (figure 4), we see link-PLSA and link-LDA fit better than the frequency-based method. This suggests that link-PLSA and link-LDA overfit due to the large number of free parameters (KD) these models have for modeling links. LTHM, on the other hand, has only D additional parameters for modeling links. Moreover, link generation probabilities depend on the topic mixtures of the documents at both ends of the link. Unlike link-PLSA and link-LDA, no values of the link parameters λ can cancel this dependency.

Figure 5 shows the performance of the four methods on the webkb test set. Once again, LTHM outperforms the other methods. The frequency-based method outperforms link-PLSA and link-LDA.

As mentioned in section 3, thanks to the symmetries in LTHM, each EM iteration is computed in time linear in the size of copus times the number of topics. Training on the webkb dataset with 20 topics took 17 hours for 600 EM iterations. Training on the smaller Wikipedia dataset with 20 topcis took 30 minutes for 300 EM iterations.

5 Discussion

In this work we have presented LTHM, a novel topic model for hypertext documents. In LTHM, the generation of hyperlinks depends on the topics of the source word and the target document of the link, as well as the relative importance of the target document. Compared to previous approaches, LTHM introduces a much smaller number of additional link parameters. As a result, LTHM achieves good generalization results in cases where other models overfit and fail to generalize.

Acknowledgements

Support from the Israeli Science Foundation is gratefully acknowledged.

References

[1] David Blei and John Lafferty. Dynamic topic models. In *International Conference on Machine Learning (ICML)*, 2006.

[2] David M. Blei and John Lafferty. Correlated topic models. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, Cambridge, MA, 2005. MIT Press.

[3] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] Wray L. Buntine and Aleks Jakulin. Applying discrete pca in data analysis. In *Uncertainty in Artificial Intelligence (UAI)*, pages 59–66, 2004.

[5] David Cohn and Huan Chang. Learning to probabilistically identify authoritative documents. In *Proc. 17th International Conf. on Machine Learning*, pages 167–174. Morgan Kaufmann, San Francisco, CA, 2000.

[6] David Cohn and Thomas Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In *Neural Information Processing Systems 13*, 2001.

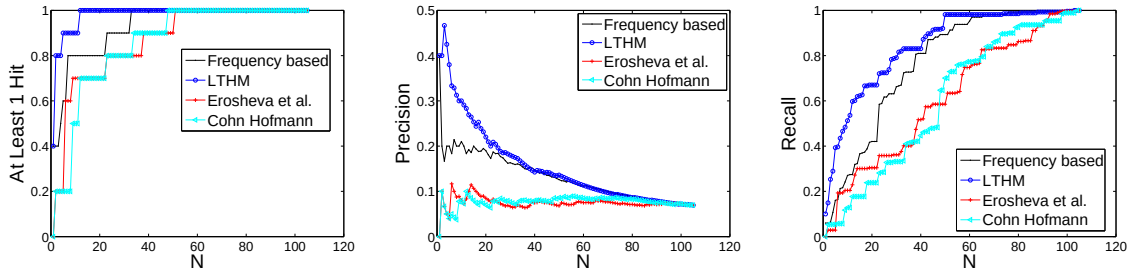
[7] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[8] Laura Dietz, Steffen Bickel, and Tobias Scheffer. Unsupervised prediction of citation influences. In *International Conference on Machine Learning (ICML)*, 2007.

[9] E. Erosheva, S. Fienberg, and J. Lafferty. Mixed-membership models of scientific publications. *Proceedings of the National Academy of Sciences*, 101:5220–5227, 2004.

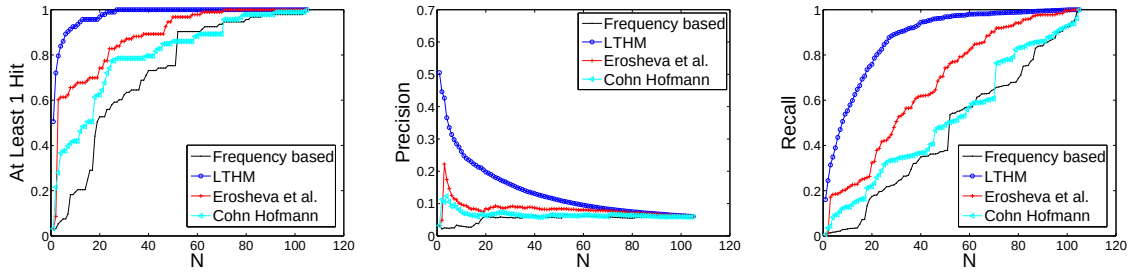
[10] Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. Integrating topics and syntax. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.

[11] Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. Integrating topics and syntax. In Lawrence K. Saul, Yair Weiss,



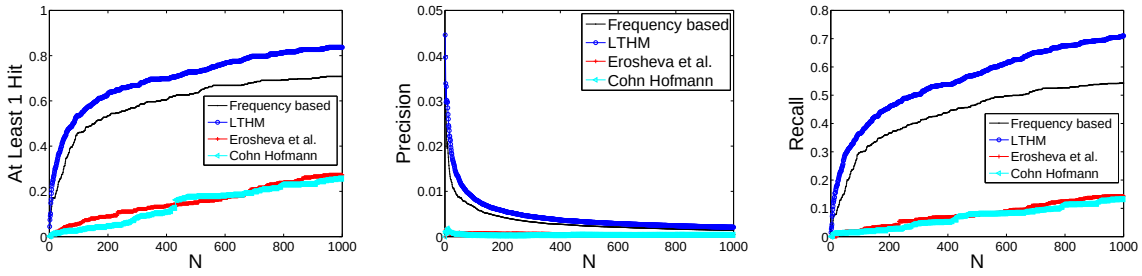
a b c

Figure 3: Comparison of link prediction in the Wikipedia test set between LTHM, link-PLSA [6], link-LDA [9] and a frequency-based method. **a.** The percentage of text documents for which there is at least one true link among the first N predicted links. **b.** Average precision for the three methods. **c.** Average recall. LTHM outperforms the other methods, while link-PLSA and link-LDA do worse than the frequency-based method, possibly due to overfitting. See figure 4 and details in the text.



a b c

Figure 4: Link prediction in the Wikipedia train set as a measure of parameter fitting by the different methods. Link-PLSA and link-LDA outperform the frequency-based method on the train set, but do worse on the test (figure 3). This suggests overfitting of these methods. **a.** The percentage of text documents for which there is at least one true link among the first N predicted links. **b.** Average precision for the three methods. **c.** Average recall.



a b c

Figure 5: Comparison of link prediction in the webkb test set between LTHM, link-PLSA, link-LDA and a frequency-based method. **a.** The percentage of text documents for which there is at least one true link among the first N predicted links. **b.** Average precision for the three methods. **c.** Average recall. LTHM outperforms the other methods.

and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 537–544. MIT Press, Cambridge, MA, 2005.

[12] Amit Gruber, Michal Rosen-Zvi, and Yair Weiss. Hidden topic markov models. In *Artificial Intelligence and Statistics (AISTATS)*, 2007.

[13] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 289–29, San Francisco, CA, 1999. Morgan Kaufmann.

[14] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[15] Thomas Minka and John Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 352–359, San Francisco, CA, 2002. Morgan Kaufmann Publishers.

[16] Ramesh Nallapati and William Cohen. Link-plsa-lda: A new unsupervised model for topics and influence of blogs. In *International Conference for Weblogs and Social Media*, 2008.

[17] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[18] Michal Rosen-Zvi, Tom Griffith, Mark Steyvers, and Padhraic Smyth. The author-topic model for authors and documents. In Max Chickering and Joseph Halpern, editors, *Proceedings 20th Conference on Uncertainty in Artificial Intelligence*, pages 487–494, San Francisco, CA, 2004. Morgam Kaufmann.

[19] Hanna M. Wallach. Topic modeling: Beyond bag-of-words. In *ICML '06: 23rd International Conference on Machine Learning*, Pittsburgh, Pennsylvania, USA, June 2006.

Appendix: Efficient Computation of $E(U_{d,z})$

Naïve computation of $E(U_{d,z})$ requires the computation of D posterior probabilities for each word in the corpus. Taking advantage of symmetries in the model, the aggregation $E(U_{d,z})$ can be computed in $O(K\sum_d N_d)$ (corpus size $\times$ number of topics):

$U_{d,z}$ is the expected number of times that $\tau_{d',i} = d$ for any d', i where $z^L_{d',i,d} = z$ but no link has been created (because $z^W_{d',i} \neq z$). Let P denote the parameters of the model, let O denote the full set of observations, and let $\hat{O}$ denote the full set of observation except for the existence or non-existence of the link under discussion. By definition,

$$E(U_{d,z}) \tag{10}$$
$$= \sum_{(d',i)\in B} \sum_{z'\neq z} \Pr(\tau_{d',i)} = d, z^L_{d',i,d} = z, z^W_{d',i} = z'|O, P)$$

where $B = \{(d', i) : no - link(d', i)\}$. It can be shown that $E(U_{d,z})$ can be written as

$$E(U_{d,z}) = \tag{11}$$
$$\lambda_d\theta_d(z)\left[\sum_{(d',i)\in B} \frac{(1 - \Pr(z^W_{d',i} = z|\hat{O}, P))}{\Pr(no - link(d', i)|\hat{O}, P)}\right]$$

The probability $\Pr(no - link(d, i)|\hat{O}, P)$ depends on the distribution $\Pr(z^W_{d,i} = z|\hat{O}, P)$. The latter one can be easily computed from the previously computed posterior distribution of topics of words given all the observations.

$$\Pr(no - link(d', i)|\hat{O}, P) \tag{12}$$
$$= 1 - \sum_z [\sum_d \lambda_d\theta_d(z)][\Pr(z^W_{d',i} = z|\hat{O}, P)]$$

To efficiently compute all the expectations $E(U_{d,z})$, one has to follow these steps:

1. Compute $\sum_d \lambda_d\theta_d(z)$ for all z in O(DK).
2. Compute $\Pr(z^W_{d',i} = z|\hat{O}, P)$ from $\Pr(z^W_{d',i} = z|O, P)$, then compute $\Pr(no - link(d', i)|\hat{O}, P)$ for all d', i in $O(K\sum_d N_d)$ (the number of topics times the size of the corpus).
3. Compute the inner brackets in equation 11 for all topics z in $O(K\sum_d N_d)$.
4. Compute $E(U_{d.z})$ for all d, z according to equation 11 in $O(KD)$.

A Game-Theoretic Analysis of Updating Sets of Probabilities

Peter D. Grünwald
CWI, P.O. Box 94079
1090 GB Amsterdam
pdg@cwi.nl

Joseph Y. Halpern
Cornell University
Ithaca, NY 14853
halpern@cs.cornell.edu

Abstract

We consider how an agent should update her uncertainty when it is represented by a set $\mathcal{P}$ of probability distributions and the agent observes that a random variable X takes on value x, given that the agent makes decisions using the *minimax criterion*, perhaps the best-studied and most commonly-used criterion in the literature. We adopt a game-theoretic framework, where the agent plays against a bookie, who chooses some distribution from $\mathcal{P}$. We consider two reasonable games that differ in what the bookie knows when he makes his choice. Anomalies that have been observed before, like *time inconsistency*, can be understood as arising because different games are being played, against bookies with different information. We characterize the important special cases in which the optimal decision rules according to the minimax criterion amount to either conditioning or simply ignoring the information. Finally, we consider the relationship between conditioning and *calibration* when uncertainty is described by sets of probabilities.

1 INTRODUCTION

Suppose that an agent models her uncertainty about a domain using a *set* $\mathcal{P}$ of probability distributions. How should the agent make decisions? Perhaps the best-studied and most commonly-used approach in the literature is to use the minimax criterion [Wald 1950; Gärdenfors and Sahlin 1982; Gilboa and Schmeidler 1989]. According to the minimax criterion, action a_1 is preferred to action a_2 if the worst-case expected loss of a_1 (with respect to all the probability distributions in the set $\mathcal{P}$ under consideration) is better than the worst-case expected loss of a_2. Thus, the action chosen is the one with the best worst-case outcome.

We are often interested in making decisions, not just in a static situation, but in a more dynamic situation, where the agent may make some observations, or learn some information. This leads to an obvious question: If the agent represents her uncertainty using a set $\mathcal{P}$ of probability distributions, how should she update $\mathcal{P}$ in light of observing that random variable X takes on value x? Perhaps the standard answer is to condition each distribution in $\mathcal{P}$ on $X = x$ (more precisely, to condition those distributions in $\mathcal{P}$ that give $X = x$ positive probability on $X = x$), and adopt the resulting set of conditional distributions $\mathcal{P} \mid X = x$ as her representation of uncertainty. As has been pointed out by several authors, this sometimes leads to a phenomenon called *dilation* [Augustin 2003; Cozman and Walley 2001; Herron, Seidenfeld, and Wasserman 1997; Seidenfeld and Wasserman 1993]: the agent may have substantial knowledge about some other random variable Y before observing $X = x$, but know significantly less after conditioning. Walley [1991, p. 299] gives a simple example of dilation: suppose that a fair coin is tossed twice, where the second toss may depend in an arbitrary way on the first. (In particular, the tosses might be guaranteed to be identical, or guaranteed to be different.) If X represents the outcome of the first toss and Y represents the outcome of the second toss, then before observing X, the agent believes that the probability that Y is heads is $1/2$, while after observing X, the agent believes that the probability that Y is heads can be an arbitrary element of $[0, 1]$.

While, as this example and others provided by Walley show, such dilation can be quite reasonable, it interacts rather badly with the minimax criterion, leading to anomalous behavior that has been called *time inconsistency* [Grünwald and Halpern 2004; Seidenfeld 2004]: the minimax-optimal conditional decision rule before the value of X is observed (which has the form "If $X = 0$ then do a_1; if $X = 1$ then do a_2; ...") may be different from the minimax decision rule obtained after conditioning. For example, the minimax-optimal conditional decision rule may say "If $X = 0$ then do a_1", but the minimax-optimal decision rule conditional on observing $X = 0$ may be a_2. (See Example 2.1.) If uncertainty is modeled using a single distribution, such time inconsistency cannot arise.

To understand this phenomenon better, we model the decision problem as a game between the agent and a bookie. It turns out that there is more than one possible game that can be considered, depending on what information the bookie has. We focus on two (closely related) games here. In the first game, the bookie chooses a distribution from $\mathcal{P}$ before the agent moves. We show that the Nash equilibrium of this game leads to a minimax decision rule. (Indeed, this can be viewed as a justification of using the minimax criterion). However, in this game, conditioning on the information is not always optimal.[1] In the second game, the bookie gets to choose the distribution *after* the value of X is observed. Again, in this game, the Nash equilibrium leads to the use of minimax, but now conditioning *is* the right thing to do.

If $\mathcal{P}$ is a singleton, the two games coincide (since there is only one choice the bookie can make, and the agent knows what it is). Not surprisingly, conditioning is the appropriate thing to do in this case. The moral of this analysis is that, when uncertainty is characterized by a set of distributions, if the agent is making decision using the minimax criterion, then the right decision depends on the game being played. The agent must consider if she is trying to protect herself against an adversary who knows the value of $X = x$ when choosing the distribution or one that does not know the value of $X = x$.

In earlier work [Grünwald and Halpern 2004] (GH from now on), we essentially considered the first game, and showed that, in this game, conditioning was not always the right thing to do when using the minimax criterion. Indeed, we showed there are sets $\mathcal{P}$ and games for which the minimax-optimal decision rule is to simply ignore the information. Our analysis of the first game lets us go beyond GH here in two ways. First, we characterize exactly when it is minimax optimal to ignore information. Second, we provide a simple sufficient condition for when conditioning on the information is minimax optimal.

Ignoring the information can be viewed as the result of conditioning; not conditioning on the information, but conditioning on the whole space. This leads to a natural question: suppose that when we observe x, we condition on the event that $X \in \mathcal{C}(x)$, where $\mathcal{C}(x)$ is some set containing x, but not necessarily equal to $\{x\}$. Is this variant of conditioning, an approach we call *$\mathcal{C}$-conditioning*, always minimax optimal in the first game? That is, is it always optimal to condition on *something*? As we show by considering the well-known Monty Hall Problem (Example 5.3), this is not the case in general. Nevertheless, $\mathcal{C}$-conditioning has some interesting properties: it is closely related to the concept of *calibration* [Dawid 1982]. Calibration is usually defined in terms of empirical data. To explain what it means, consider an agent that is a weather forecaster on your local television station. Every night the forecaster makes a prediction about whether or not it will rain the next day in the area where you live. She does this by asserting that the probability of rain is p, where $p \in \{0, 0.1, \ldots, 0.9, 1\}$. How should we interpret these probabilities? The usual interpretation is that, in the long run, on those days at which the weather forecaster predict probability p, it will rain approximately $100p\%$ of the time [Dawid 1982]. Thus, for example, among all days for which she predicted 0.1, the fraction of days with rain was close to 0.1. A weather forecaster with this property is called *calibrated*.

Up to now, calibration has been considered only when uncertainty is characterized by a single distribution. We generalize the notion of calibration to our setting, where uncertainty is characterized by a set of distributions. We then show that a rule for updating a set of probabilities is guaranteed to be calibrated if and only if it is an instance of $\mathcal{C}$-conditioning. In combination with our earlier results, this implies that if calibration is considered essential, then an update rule may sometimes result in decisions that are not minimax optimal.

Both the idea of representing uncertainty by a set $\mathcal{P}$ of distributions and that of handling decisions in a worst-case optimal manner may, of course, be criticized. While we do not claim that this is necessarily the "right" or the "best" approach, two of the most common criticisms are, to some extent, unjustified. First, since it may be hard for an agent to determine the precise boundaries of the set $\mathcal{P}$, it has been argued that "soft boundaries" are more appropriate. While this is sometimes the case, hard boundaries are natural in some cases, such as the Monty Hall problem (Example 5.3). Similarly, the use of the minimax criterion is not as pessimistic as is often thought. The minimax solution often coincides with the Bayes-optimal solution under some "maximum entropy" prior [Grünwald and Dawid 2004], which is not commonly associated with being overly pessimistic. In fact, in the Monty Hall problem, the minimax-optimal decision rule coincides with the solution usually advocated, which requires making further assumptions about $\mathcal{P}$ to reduce it to a singleton.

2 NOTATION AND DEFINITIONS

Preliminaries: For ease of exposition, we assume throughout this paper that we are interested in two random variables, X and Y, which can take values in spaces $\mathcal{X}$ and $\mathcal{Y}$, respectively. $\mathcal{P}$ always denotes a set of distributions on $\mathcal{X} \times \mathcal{Y}$; that is, $\mathcal{P} \subseteq \Delta(\mathcal{X} \times \mathcal{Y})$, where, as usual, $\Delta(S)$ denotes the set of probability distributions on S. For ease of exposition, we assume that $\mathcal{P}$ is a closed set; this is a standard assumption in the literature that seems quite natural in our applications, and makes the statement of our results simpler. If $\Pr \in \Delta(\mathcal{X} \times \mathcal{Y})$, let $\Pr_{\mathcal{X}}$ and $\Pr_{\mathcal{Y}}$ denote the marginals of $\Pr$ on $\mathcal{X}$ and $\mathcal{Y}$, respectively.

[1] In some other senses of the words "conditioning" and "optimal," conditioning on the information *is* always optimal. This is discussed further in Section 6.

Let $\mathcal{P}_\mathcal{Y} = \{\Pr_\mathcal{Y} : \Pr \in \mathcal{P}\}$. If $E \subseteq \mathcal{X} \times \mathcal{Y}$, then let $\mathcal{P} \mid E = \{\Pr \mid E : \Pr \in \mathcal{P}, \Pr(E) > 0\}$. Here $\Pr \mid E$ (denoted by some authors as $\Pr(\cdot \mid E)$) is the distribution on $\mathcal{X} \times \mathcal{Y}$ obtained by conditioning on E.

Loss Functions: As in GH, we are interested in an agent who must choose some action from a set $\mathcal{A}$, where the loss of the action depends only on the value of random variable Y. For ease of exposition, we assume in this paper that $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{A}$ are always finite. We assume that with each action $a \in \mathcal{A}$ and value $y \in \mathcal{Y}$ is associated some loss to the agent. (The losses can be negative, which amounts to a gain.) Let $L : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}$ be the loss function.[2]

Such loss functions arise quite naturally. For example, in a medical setting, we can take $\mathcal{Y}$ to consist of the possible diseases and $\mathcal{X}$ to consist of symptoms. The set $\mathcal{A}$ consists of possible courses of treatment that a doctor can choose. The doctor's loss function depends only on the patient's disease and the course of treatment, not on the symptoms. But, in general, the doctor's choice of treatment depends on the symptoms observed.

Decision Rules: Suppose that the agent observes the value of a variable X that takes on values in $\mathcal{X}$. After having observed X, she must perform an act, the quality of which is judged according to loss function L. The agent must choose a *decision rule* that determines what she does as a function of her observations. We allow decision rules to be randomized. Thus, a decision rule is a function $\delta : \mathcal{X} \to \Delta(\mathcal{A})$ that chooses a distribution over actions based on the agent's observations. Let $\mathcal{D}(\mathcal{X}, \mathcal{A})$ be the set of all decision rules. A special case is a deterministic decision rule, which assigns probability 1 to a particular action. If δ is deterministic, we sometimes abuse notation and write $\delta(x)$ for the action that is assigned probability 1 by the distribution $\delta(x)$. Given a decision rule δ and a loss function L, let L_δ be the random variable on $\mathcal{X} \times \mathcal{Y}$ such that $L_\delta(x, y) = \sum_{a \in \mathcal{A}} \delta(x)(a) L(y, a)$. Here $\delta(x)(a)$ stands for the probability of performing action a according to the distribution $\delta(x)$ over actions that is adopted when x is observed. Note that in the special case that δ is a deterministic decision rule, $L_\delta(x, y) = L(y, \delta(x))$.

A decision rule δ^0 is *a priori minimax optimal* with respect to $\mathcal{P}$ and $\mathcal{A}$ if

$$\max_{\Pr \in \mathcal{P}} E_{\Pr}[L_{\delta^0}] = \min_{\delta \in \mathcal{D}(\mathcal{X}, \mathcal{A})} \max_{\Pr \in \mathcal{P}} E_{\Pr}[L_\delta].$$

That is, δ^0 is a priori minimax optimal if δ^0 gives the best worst-case expected loss with respect to all the distributions in $\Pr$. We can write max here instead of sup because of our assumption that $\mathcal{P}$ is closed. This ensures that there is some $\Pr \in \mathcal{P}$ for which $E_{\Pr}[L_{\delta^0}]$ takes on its maximum value.

[2] We could equally well use utilities, which can be viewed as a positive measure of gain. Losses seem to be somewhat more standard in this literature.

A decision rule δ^1 is *a posteriori minimax optimal* with respect to $\mathcal{P}$ and $\mathcal{A}$ if, for all $x \in \mathcal{X}$ such that $\Pr(X = x) > 0$ for some $\Pr \in \mathcal{P}$,

$$\max_{\Pr \in \mathcal{P} \mid X = x} E_{\Pr}[L_{\delta^1}] = \min_{\delta \in \mathcal{D}(\mathcal{X}, \mathcal{A})} \max_{\Pr \in \mathcal{P} \mid X = x} E_{\Pr}[L_\delta]. \quad (1)$$

To get the a posteriori minimax-optimal decision rule we do the obvious thing: if x is observed, we simply condition each probability distribution $\Pr \in \mathcal{P}$ on $X = x$, and choose the action that gives the least expected loss (in the worst case) with respect to $\mathcal{P} \mid X = x$. Since all distributions $\Pr$ mentioned in (1) satisfy $\Pr(X = x) = 1$, the minimum over $\delta \in \mathcal{D}(\mathcal{X}, \mathcal{A})$ does not depend on the values of $\delta(x')$ for $x' \neq x$; the minimum is effectively over randomized actions rather than decision rules.

As the following example, taken from GH, shows, a priori minimax-optimal decision rules are in general different from a posteriori minimax-optimal decision rules.

Example 2.1: Suppose that $\mathcal{X} = \mathcal{Y} = \mathcal{A} = \{0, 1\}$ and $\mathcal{P} = \{\Pr \in \Delta(\mathcal{X} \times \mathcal{Y}) : \Pr_\mathcal{Y}(Y = 1) = 2/3\}$. Thus, $\mathcal{P}$ consists of all distributions whose marginal on Y gives $Y = 1$ probability $2/3$. We can think of the actions in $\mathcal{A}$ as predictions of the value of Y. The loss function is 0 if the right value is predicted and 1 otherwise; that is, $L(i, j) = |i - j|$. This is the so-called $0/1$ or *classification* loss. It is easy to see that the optimal a priori decision rule is to choose 1 no matter what is observed (which has expected loss $1/3$). Intuitively, observing the value of X tells us nothing about the value of Y, so the best decision is to predict according to the prior probability of $Y = 1$. However, all probabilities on $Y = 1$ are compatible with observing either $X = 0$ or $X = 1$. That is, both $(\mathcal{P} \mid X = 0)_\mathcal{Y}$ and $(\mathcal{P} \mid X = 1)_\mathcal{Y}$ consist of all distributions on $\mathcal{Y}$. Thus, the minimax optimal a posteriori decision rule randomizes (with equal probability) between $Y = 0$ and $Y = 1$.

Thus, if you make decisions according to the minimax rule, then before making an observation, you will predict $Y = 1$. However, *no matter what observation you make*, after making the observation, you will randomize (with equal probability) between predicting $Y = 0$ and $Y = 1$. Moreover, you know even before making the observation that your opinion as to the best decision rule will change in this way. ∎

3 TWO GAME-THEORETIC INTERPRETATIONS OF $\mathcal{P}$

What does it mean that an agent's uncertainty is characterized by a set $\mathcal{P}$ of probability distributions? How should we understand $\mathcal{P}$? We give $\mathcal{P}$ a game-theoretic interpretation here: namely, an adversary gets to choose a distribu-

tion from the set $\mathcal{P}$.[3] But this does not completely specify the game. We must also specify *when* the adversary makes the choice. We consider two times that the adversary can choose: the first is before the agents observes the value of $\mathcal{X}$, and the second is after. We formalize this as two different games, where we take the "adversary" to be a bookie.

We call the first game the $\mathcal{P}$-game. It is defined as follows:

1. The bookie chooses a distribution $\Pr \in \mathcal{P}$.
2. The value x of X is chosen (by nature) according to $\Pr_{\mathcal{X}}$ and observed by both bookie and agent.
3. The agent chooses an action $a \in \mathcal{A}$.
4. The value y of Y is chosen according to $\Pr \mid X = x$.
5. The agent's loss is $L(y, a)$; the bookie's loss is $-L(y, a)$.

This is a zero-sum game; the agent's loss is the bookie's gain. In this game, the agent's strategy is a decision rule, that is, a function that gives a distribution over actions for each observed value of X. The bookie's strategy is a distribution over distributions in $\mathcal{P}$.

We now consider a second interpretation of $\mathcal{P}$, characterized by a different game that gives the bookie more power. Rather than choosing the distribution before observing the value of X, the bookie gets to choose the distribution after observing the value. We call this the $\mathcal{P}$-X-game.

1. The value x of X is chosen (by nature) in such a way that $\Pr(X = x) > 0$ for some $\Pr \in \mathcal{P}$, and observed by both the bookie and the agent.
2. The bookie chooses a distribution $\Pr \in \mathcal{P}$ such that $\Pr(X = x) > 0$.[4]
3. The agent chooses an action $a \in \mathcal{A}$.
4. The value y of Y is chosen according to $\Pr \mid X = x$.
5. The agent's loss is $L(y, a)$; the bookie's loss is $-L(y, a)$.

Recall that a pair of strategies (S_1, S_2) is a Nash equilibrium if neither party can do better by unilaterally changing strategies. If, as in our case, (S_1, S_2) is a Nash equilibrium in a zero-sum game, it is also known as a "saddle point"; S_1 must be a minimax strategy, and S_2 must be a maximin strategy [Grünwald and Dawid 2004]. As the following results show, an agent must be using an a priori minimax-optimal decision rule in a Nash equilibrium of the $\mathcal{P}$-game, and an a posteriori minimax-optimal decision rule is a Nash equilibrium of the $\mathcal{P}$-X-game. This can be viewed as a justification for using (a priori and a posteriori) minimax-optimal decision rules.

[3] This interpretation remains meaningful in several practical situations where there is no explicit adversary; see the final paragraph of this section.

[4] If we were to consider conditional probability measures, for which $\Pr(Y = y \mid X = x)$ is defined even if $\Pr(X = x) = 0$, then we could drop the restriction that x is chosen such that $\Pr(X = x) > 0$ for some $\Pr \in \mathcal{P}$.

Theorem 3.1: *Fix $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{A}$, L, and $\mathcal{P} \subseteq \Delta(\mathcal{X} \times \mathcal{Y})$.*

(a) The $\mathcal{P}$-game has a Nash equilibrium (π^*, δ^*), where π^* is a distribution over $\mathcal{P}$ with finite support.

(b) If (π^*, δ^*) is a Nash equilibrium of the $\mathcal{P}$-game such that π^* has finite support, then

(i) for every distribution $\Pr' \in \mathcal{P}$ in the support of π^*, we have $E_{\Pr'}[L_{\delta^*}] = \max_{\Pr \in \mathcal{P}} E_{\Pr}[L_{\delta^*}]$;

(ii) if $\Pr^* = \sum_{\Pr \in \mathcal{P}, \pi^*(\Pr) > 0} \pi^*(\Pr) \Pr$ (i.e., $\Pr^*$ is the convex combination of the distributions in the support of π^*, weighted by their probability according to π^*), then

$$\begin{aligned} E_{\Pr^*}[L_{\delta^*}] &= \min_{\delta \in \mathcal{D}(\mathcal{X}, \mathcal{A})} E_{\Pr^*}[L_\delta] \\ &= \max_{\Pr \in \mathcal{P}} \min_{\delta \in \mathcal{D}(\mathcal{X}, \mathcal{A})} E_{\Pr}[L_\delta] \\ &= \min_{\delta \in \mathcal{D}(\mathcal{X}, \mathcal{A})} \max_{\Pr \in \mathcal{P}} E_{\Pr}[L_\delta] \\ &= \max_{\Pr \in \mathcal{P}} E_{\Pr}[L_{\delta^*}]. \end{aligned}$$

Once nature has chosen a value for X in the $\mathcal{P}$-X-game, we can regard steps 2–5 of the $\mathcal{P}$-X-game as a game between the bookie and the agent, where the bookie's strategy is characterized by a distribution in $\mathcal{P} \mid X = x$ and the agent's is characterized by a distribution over actions. We call this the $\mathcal{P}$-x-game.

Theorem 3.2: *Fix $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{A}$, L, $\mathcal{P} \subseteq \Delta(\mathcal{X} \times \mathcal{Y})$.*

(a) The $\mathcal{P}$-x-game has a Nash equilibrium $(\pi^*, \delta^*(x))$, where π^* is a distribution over $\mathcal{P} \mid X = x$ with finite support.

(b) If $(\pi^*, \delta^*(x))$ is a Nash equilibrium of the $\mathcal{P}$-x-game such that π^* has finite support, then

(i) for all $\Pr'$ in the support of π^*, we have $E_{\Pr'}[L_{\delta^*}] = \max_{\Pr \in \mathcal{P} \mid X = x} E_{\Pr}[L_{\delta^*}]$;

(ii) if $\Pr^* = \sum_{\Pr \in \mathcal{P}, \pi^*(\Pr) > 0} \pi^*(\Pr) \Pr$, then

$$\begin{aligned} E_{\Pr^*}[L_{\delta^*}] &= \min_{\delta \in \mathcal{D}(\mathcal{X}, \mathcal{A})} E_{\Pr^*}[L_\delta] \\ &= \max_{\Pr \in \mathcal{P} \mid X = x} \min_{\delta \in \mathcal{D}(\mathcal{X}, \mathcal{A})} E_{\Pr}[L_\delta] \\ &= \min_{\delta \in \mathcal{D}(\mathcal{X}, \mathcal{A})} \max_{\Pr \in \mathcal{P} \mid X = x} E_{\Pr}[L_\delta] \\ &= \max_{\Pr \in \mathcal{P} \mid X = x} E_{\Pr}[L_{\delta^*}]. \end{aligned}$$

Since all distributions $\Pr$ in the expression $\min_{\delta \in \mathcal{D}(\mathcal{X}, \mathcal{A})} \max_{\Pr \in \mathcal{P} \mid X = x} E_{\Pr}[L_\delta]$ in part (b)(ii) are in $\mathcal{P} \mid X = x$, as in (1), the minimum is effectively over randomized actions rather than decision rules.

The proof of Theorems 3.1 and 3.2, as well as all other missing proofs, can be found in the full paper [Grünwald and Halpern 2007]. These theorems can be viewed as saying that there is no time inconsistency; rather, we must just be careful about what game is being played. If the $\mathcal{P}$-game is being played, the right strategy is the a priori minimax-optimal strategy, both before and after the value of X is observed; similarly, if the $\mathcal{P}$-X-game is being played, the

right strategy is the a posteriori minimax-optimal strategy, both before and after the value of X is observed. Indeed, thinking in terms of the games explains the apparent time inconsistency. While it is true that the agent gains more information by observing $X = x$, in the $\mathcal{P}$-X game, so does the bookie. This information may be of more use to the bookie than the agent, so, in this game, the agent can be worse off by being given the opportunity to learn the value of X.

Of course, in most practical situations, agents (robots, statisticians,...) are not really confronted with a bookie who tries to make them suffer. Rather, the agents may have no idea at all what distribution holds, except that it is in some set $\mathcal{P}$. Because all they know is $\mathcal{P}$, they decide to prepare themselves for the worst-case and play the minimax strategy. The fact that such a minimax strategy can be interpreted in terms of a Nash equilibrium of a game helps to understand differences between different forms of minimax (such as a priori and a posteriori minimax). From this point of view, it seems strange to have a bookie choose between different distributions in $\mathcal{P}$ according to some distribution π^*. However, if $\mathcal{P}$ is convex, we can replace the distribution π^* on $\mathcal{P}$ by a single distribution in $\mathcal{P}$, which consists of the convex combination of the distributions in the support of π^*; this is just the distribution $\Pr^*$ of Theorems 3.1 and 3.2. Thus, Theorems 3.1 and 3.2 hold with the bookie restricted to a deterministic strategy.

4 CHARACTERIZING A PRIORI MINIMAX DECISION RULES

To get the a posteriori minimax-optimal decision rule we do the obvious thing: if x is observed, we simply condition each probability distribution $\Pr \in \mathcal{P}$ on $X = x$, and choose the action that gives the least expected loss (in the worst case) with respect to $\mathcal{P} \mid X = x$.

We might expect that the a priori minimax-optimal decision rule should do the same thing. That is, it should be the decision rule that says, if x is observed, then we choose the action that again gives the best result (in the worst case) with respect to $\mathcal{P} \mid X = x$. However, as shown in GH, this intuition is incorrect in general. There are times, for example, that the best thing to do is to ignore the observed value of X, and just choose the action that gives the least expected loss (in the worst case) with respect to $\mathcal{P}$, no matter what value X has. In this section we first give a sufficient condition for conditioning to be optimal, and then characterize when ignoring the observed value is optimal.

Definition 4.1 : Let $\langle \mathcal{P} \rangle = \{\Pr \in \Delta(\mathcal{X} \times \mathcal{Y}) : \Pr_{\mathcal{X}} \in \mathcal{P}_{\mathcal{X}}$ and $(\Pr \mid X = x) \in (\mathcal{P} \mid X = x)$ for all $x \in \mathcal{X}$ such that $\mathcal{P} \mid X = x$ is nonempty$\}$. ∎

Thus, $\langle \mathcal{P} \rangle$ consists of all distributions $\Pr$ whose marginal on $\mathcal{X}$ is the marginal on $\mathcal{X}$ of some distribution in $\mathcal{P}$ and whose conditional on observing $X = x$ is the conditional of some distribution in $\mathcal{P}$, for all $x \in \mathcal{X}$. Clearly $\mathcal{P} \subseteq \langle \mathcal{P} \rangle$, but the converse is not necessarily true. When it is true, conditioning is optimal.

Proposition 4.2: *If $\mathcal{P} = \langle \mathcal{P} \rangle$, then there exists an a priori minimax-optimal rule that is also a posteriori minimax optimal. If, for all $\Pr \in \mathcal{P}$ and all $x \in \mathcal{X}$, $\Pr(X = x) > 0$, then every a priori minimax-optimal rule is also a posteriori minimax optimal.*

As we saw in Example 2.1, the minimax-optimal a priori decision rule is not always the same as the minimax-optimal a posteriori decision rule. In fact, the minimax-optimal a priori decision rule ignores the information observed. Formally, a rule δ *ignores information* if $\delta(x) = \delta(x')$ for all $x, x' \in \mathcal{X}$. If δ ignores information, define L'_δ to be the random variable on $\mathcal{Y}$ such that $L'_\delta(y) = L_\delta(x, y)$ for some choice of x. This is well defined, since $L_\delta(x, y) = L_\delta(x', y)$ for all $x, x' \in \mathcal{X}$.

Theorem 4.3: *Fix $\mathcal{X}$, $\mathcal{Y}$, L, $\mathcal{A}$, and $\mathcal{P} \subseteq \Delta(\mathcal{X} \times \mathcal{Y})$. If, for all $\Pr_{\mathcal{Y}} \in \mathcal{P}_{\mathcal{Y}}$, $\mathcal{P}$ contains a distribution $\Pr'$ such that X and Y are independent under $\Pr'$, and $\Pr'_{\mathcal{Y}} = \Pr_{\mathcal{Y}}$, then there is an a priori minimax-optimal decision rule that ignores information. Under these conditions, if δ is an a priori minimax-optimal decision rule that ignores information, then δ essentially optimizes with respect to the marginal on Y; that is,* $\max_{\Pr \in \mathcal{P}} E_{\Pr}[L_\delta] = \max_{\Pr_{\mathcal{Y}} \in \mathcal{P}_{\mathcal{Y}}} E_{\Pr_{\mathcal{Y}}}[L'_\delta]$.

GH focused on the case that $\mathcal{P}_{\mathcal{Y}}$ is a singleton (i.e., the marginal probability on Y is the same for all distributions in $\mathcal{P}$) and for all x, $\mathcal{P}_{\mathcal{Y}} \subseteq (\mathcal{P} \mid X = x)_{\mathcal{Y}}$. It is immediate from Theorem 4.3 that ignoring information is a priori minimax optimal in this case.

5 $\mathcal{C}$-CONDITIONING & CALIBRATION

Conditioning is the most common way of updating uncertainty. In this section, we examine updating by conditioning. The following definition makes precise the idea that a decision rule is based on conditioning.

Definition 5.1 : A *probability update rule* is a function $\Pi : 2^{\Delta(\mathcal{X} \times \mathcal{Y})} \times \mathcal{X} \to 2^{\Delta(\mathcal{X} \times \mathcal{Y})}$ mapping a set $\mathcal{P}$ of distributions and an observation x to a set $\Pi(\mathcal{P}, x)$ of distributions; intuitively, $\Pi(\mathcal{P}, x)$ is the result of updating $\mathcal{P}$ with the observation x. ∎

Definition 5.2: Let $\mathcal{C} = \{\mathcal{X}_1, \ldots, \mathcal{X}_k\}$ be a partition of $\mathcal{X}$; that is, $\mathcal{X}_i \neq \emptyset$ for $i = 1, \ldots, k$; $\mathcal{X}_1 \cup \ldots \mathcal{X}_k = \mathcal{X}$; and $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ for $i \neq j$. If $x \in \mathcal{X}$, let $\mathcal{C}(x)$ be the cell containing x; i.e., the unique element $\mathcal{X}_i \in \mathcal{C}$ such that $x \in \mathcal{X}_i$. The *$\mathcal{C}$-conditioning* probability update rule is the function Π defined by taking $\Pi(\mathcal{P}, x) = \mathcal{P} \mid X \in$

$\mathcal{C}(x)$. A decision rule δ is *based on $\mathcal{C}$-conditioning* if it amounts to first updating the set $\mathcal{P}$ to $\mathcal{P} \mid X \in \mathcal{C}(x)$, and then taking the minimax-optimal distribution over actions relative to $\mathcal{P} \mid X \in \mathcal{C}(x)$. Formally, δ is based on $\mathcal{C}$-conditioning if, for all $x \in \mathcal{X}$ with $\Pr(X = x) > 0$ for some $\Pr \in \mathcal{P}$,

$$\max_{\Pr \in \mathcal{P} | X \in \mathcal{C}(x)} E_{\Pr}[L_\delta] = \min_{\delta \in \mathcal{D}(\mathcal{X}, \mathcal{A})} \max_{\Pr \in \mathcal{P} | X \in \mathcal{C}(x)} E_{\Pr}[L_\delta].$$

∎

All examples of a priori minimax decision rules that we have seen so far are based on $\mathcal{C}$-conditioning: Standard conditioning is based on $\mathcal{C}$-conditioning, where we take $\mathcal{C}$ to consist of all singletons; ignoring information is also based on $\mathcal{C}$-conditioning, where $\mathcal{C} = \{\mathcal{X}\}$. This suggests that, perhaps, the a priori minimax decision rule must also be based on $\mathcal{C}$-conditioning. The following well-known example shows that this conjecture is false.

Example 5.3 : [The Monty Hall Problem] [Mosteller 1965; vos Savant 1990]: Suppose that you're on a game show and given a choice of three doors. Behind one is a car; behind the others are goats. You pick door 1. Before opening door 1, Monty Hall, the host (who knows what is behind each door) opens one of the other two doors, say, door 3, which has a goat. He then asks you if you still want to take what's behind door 1, or to take what's behind door 2 instead. Should you switch? You may assume that initially, the car was equally likely to be behind each of the doors.

We formalize this well-known problem as a $\mathcal{P}$-game, as follows: $\mathcal{Y} = \{1, 2, 3\}$ represents the door which the car is behind. $\mathcal{X} = \{G_2, G_3\}$, where, for $j \in \{2, 3\}$, G_j corresponds to the quizmaster showing that there is a goat behind door j. $\mathcal{A} = \{1, 2, 3\}$, where action $a \in \mathcal{A}$ corresponds to the door you finally choose, after Monty has opened door 2 or 3. The loss function is once again the classification loss, $L(i, j) = 1$ if $i \neq j$, that is, if you choose a door with a goat behind it, and $L(i, j) = 0$ if $i = j$, that is, if you choose a door with a car. $\mathcal{P}$ is the set of all distributions $\Pr$ on $\mathcal{X} \times \mathcal{Y}$ satisfying

$$\Pr_{\mathcal{Y}}(Y = 1) = \Pr_{\mathcal{Y}}(Y = 2) = \Pr_{\mathcal{Y}}(Y = 3) = \tfrac{1}{3}$$
$$\Pr(Y = 2 \mid X = G_2) = \Pr(Y = 3 \mid X = G_3) = 0.$$

It is well known, and easy to show, that the minimax-optimal strategy is always to switch doors, no matter whether Monty opens door 2 or door 3. Since the game is an instance of the $\mathcal{P}$-game, this means that the decision rule δ^* given by $\delta^*(G_2) = 3$; $\delta^*(G_3) = 2$ is an a priori minimax rule. Clearly, δ^* is *not* based on $\mathcal{C}$-conditioning: there exist only two partitions of $\mathcal{X}$. The corresponding two update rules based on $\mathcal{C}$-conditioning amount to, respectively, (a) ignoring X and choosing each door with probability 1/3, or (b) conditioning on X in the standard way and thus choosing each of the two remaining doors with probability 1/2. Neither strategy (a) nor (b) is minimax optimal. Thus, the a priori minimax decision rule in the $\mathcal{P}$-game is not always based on $\mathcal{C}$-conditioning. ∎

While the example shows that $\mathcal{C}$-conditioning is not always optimal in the minimax sense, it can be justified by other means; as we now show, $\mathcal{C}$-conditioning is closely related to *calibration*. Indeed, a probability update rule is calibrated if and only if for each $\mathcal{P}$, it amounts to $\mathcal{C}$-conditioning for some partition $\mathcal{C}$ of $\mathcal{X}$. Calibration is usually defined in terms of empirical data. To explain what it means, consider a weather forecaster, who predicts the probability of rain every day. How should we interpret the probabilities that she announces? The usual interpretation—which coincides with most people's intuitive understanding—is that, in the long run, on those days at which the weather forecaster predict probability p, it will rain approximately $100p\%$ of the time [Dawid 1982]. Thus, for example, among all days for which she predicted 0.1, the fraction of days with rain was close to 0.1 (given the weather forecaster's precision, we should require it to be between, say, 0.05 and 0.15). A weather forecaster with this property is said to be *calibrated*. If a weather forecaster is calibrated, and you make bets based on her probabilistic predictions (which are all accepted), then in the long run you will not lose money.

If a weather forecaster is not calibrated, there exist bets which seem favorable but which result in a loss. Note that calibration is a *minimal* requirement: a weather forecaster who predicts 0.3 for every single day of the year may be calibrated if it indeed rains on 30% of the days, but still not very informative. Thus, given two calibrated forecasters, we prefer the one that makes "sharper" predictions, in a sense to be defined below.

In our case, we do not test probabilistic predictions with respect to empirical relative frequencies, but with respect to other sets of "potentially underlying" probability measures. We are not the first to do this; see, for example, [Vovk, Gammerman, and Shafer 2005]. The definition of calibration extends naturally to this situation. To see how, we first define calibration with respect to a single underlying probability measure. Let $\mathcal{P} = \{\Pr\}$ for a single distribution $\Pr$ and let Π be a probability update rule (Definition 5.1) such that $\Pi(\{\Pr\}, x)$ contains just a single distribution for each $x \in \mathcal{X}$ (for example, Π could be ordinary conditioning). We define

$$\mathbf{R} = \{\mathcal{R} : \mathcal{R} = (\ \Pi(\mathcal{P}, x)\ )_{\mathcal{Y}} \text{ for some } x \in \mathcal{X}\}. \quad (2)$$

$\mathbf{R}$ is just the range of Π, restricted to distributions of Y, the random variable that we are interested in predicting; its elements are the distributions on Y that $\Pr$ is mapped to, upon observing different values of x. Note that $\mathbf{R}$ is defined relative to a probability update rule Π and a set $\mathcal{P}$ of distributions. By our assumptions on $\mathcal{P}$ and Π,

$\mathcal{R} = \{\{R_1\}, \{R_2\}, \ldots\}$ is a set of singleton sets, each containing one distribution on $\mathcal{Y}$. For $\{R\} \in \mathbf{R}$, let $\mathcal{X}_R$ be the set of $x \in \mathcal{X}$ that map $\Pr$ to R, i.e.

$$\mathcal{X}_R = \{x \in \mathcal{X} \ : \ (\ \Pi(\{\Pr\}, x)\ )_{\mathcal{Y}} = \{R\}\}.$$

Note that the sets $\{\mathcal{X}_R\}$ partition $\mathcal{X}$. Π is calibrated relative to $\mathcal{P}$ if for all R with $\{R\} \in \mathbf{R}$, $(\Pr \mid X \in \mathcal{X}_R)_{\mathcal{Y}} = R$. Thus, conditioned on the event that the agent predicts Y using distribution R, the distribution of Y must indeed be equal to R.

It is straightforward to generalize this notion to sets $\mathcal{P}$ of probability distributions that are not singletons, and update rules Π that map to sets of probabilities. Definition (2) remains unchanged. For $\mathcal{R} \in \mathbf{R}$, we now take $\mathcal{X}_{\mathcal{R}}$ to be the set of $x \in \mathcal{X}$ that map $\mathcal{P}$ to $\mathcal{R}$, that is,

$$\mathcal{X}_{\mathcal{R}} = \{x \in \mathcal{X} \ : \ (\ \Pi(\mathcal{P}, x)\ )_{\mathcal{Y}} = \mathcal{R}\}. \qquad (3)$$

Once again, the sets $\{\mathcal{X}_{\mathcal{R}}\}$ partition $\mathcal{X}$.

Definition 5.4: Π is *calibrated relative to* $\mathcal{P}$ if for all $\Pr \in \mathcal{P}$ and $\mathcal{R} \in \mathbf{R}$, $\Pr_Y(\cdot \mid X \in \mathcal{X}_{\mathcal{R}}) \in \mathcal{R}$.
Π *is calibrated* if it is calibrated relative to all sets of distributions $\mathcal{P} \subseteq \Delta(\mathcal{X} \times \mathcal{Y})$. ∎

Proposition 5.5: *For all partitions $\mathcal{C}$ of $\mathcal{X}$ and all $\mathcal{P}$, $\mathcal{C}$-conditioning is calibrated relative to $\mathcal{P}$.*

Calibration as defined here is a very weak notion. For example, the update rule $\Pi(\mathcal{P}, x) = \Delta(\mathcal{X} \times \mathcal{Y})$ that maps each combination of x and $\mathcal{P}$ to the set of all distributions on $\mathcal{X} \times \mathcal{Y}$ is calibrated under our definition. This update rule loses whatever information may have been contained in $\mathcal{P}$, and is therefore not very useful. Intuitively, the fewer distributions that there are in $\mathcal{P}$, the more information $\mathcal{P}$ contains. Thus, we restrict ourselves to sets $\mathcal{P}$ that are as small as possible, while still being calibrated.

Definition 5.6: Update rule Π' is *wider than update rule* Π *relative to* $\mathcal{P}$ if, for all $x \in \mathcal{X}$, $\Pi(\mathcal{P}, x) \subseteq \Pi'(\mathcal{P}, x)$.
Π' is *strictly wider* relative to $\mathcal{P}$ if the inclusion is strict for some some x. Π is *(strictly) narrower* than Π', relative to $\mathcal{P}$ if Π' is (strictly) wider than Π relative to $\mathcal{P}$. Π is *sharply calibrated* relative to $\mathcal{P}$ if Π is calibrated relative to $\mathcal{P}$ and there is no update rule Π' that is calibrated and strictly narrower than Π relative to $\mathcal{P}$. Π is *sharply calibrated* if Π is sharply calibrated relative to all $\mathcal{P} \subseteq \Delta(\mathcal{X} \times \mathcal{Y})$. ∎

We now want to prove that every sharply calibrated update rule must involve conditioning. To make this precise, we need the following definition.

Definition 5.7: Π is a *generalized conditioning update rule* if, for all $\mathcal{P} \subseteq \Delta(\mathcal{X} \times \mathcal{Y})$, there exists a partition $\mathcal{C}$ (that may depend on $\mathcal{P}$) such that for all $x \in \mathcal{X}$, $\Pi(\mathcal{P}, x) = \mathcal{P} \mid C(x)$. ∎

Note that in a generalized conditioning rule, we condition on a partition of $\mathcal{X}$, but the partition may depend on the set $\mathcal{P}$. For example, for some $\mathcal{P}$, the rule may ignore the value of x, whereas for other $\mathcal{P}$, it may amount to ordinary conditioning. It easily follows from Proposition 4.2 that every generalized conditioning rule is calibrated. The next result shows that every *sharply* calibrated update rule must be a generalized conditioning rule.

Theorem 5.8: *There exists an update rule that is sharply calibrated. Moreover, every sharply calibrated update rule is a generalized conditioning update rule.*

Theorem 5.8 says that an agent who wants to be sharply calibrated should update her probabilities using conditioning (although what she conditions on may depend on the set of probabilities that she considers possible).

Given the game-theoretic interpretation of Section 3, we might wonder if there is a variant of the games considered earlier for which the equilibrium involves generalized conditioning. As we show in the full paper, there is (although the game is perhaps not as natural as the ones considered in Section 3). Roughly speaking, we consider a three-player game, with a bookie and two agents. The bookie again chooses a probability distribution from a set $\mathcal{P}$; the bookie also chooses the loss function from some set. The first agent observes $\mathcal{P}$ and x and updates $\mathcal{P}$ to $\mathcal{P}_x$. The second agent learns $\mathcal{P}_x$ and b (but not $\mathcal{P}$ and x) and makes the minimax-optimal decision. As we show, in Nash equilibrium, the first agent's updated set of probabilities, $\mathcal{P}_x$, must be the result of $\mathcal{C}$-conditioning, where, as in Theorem 5.8, $\mathcal{C}$ may depend on $\mathcal{P}$.

6 DISCUSSION

We have examined how to update uncertainty represented by a set of probability distributions, where we motivate updating rules in terms of the minimax criterion. Our key innovation has been to show how different approaches can be understood in terms of a game between a bookie and an agent, where the bookie picks a distribution from the set and the agent chooses an action after making an observation. Different approaches to updating arise depending on whether the bookie's choice is made before or after the observation. We believe that this game-theoretic approach should prove useful more generally in understanding different approaches to updating. We hope to explore this further in future work.

We end this paper by giving an overview of the senses in which conditioning is optimal and the senses in which it is not, when uncertainty is represented by a set of distributions. We have established that conditioning the full set $\mathcal{P}$ on $X = x$ is minimax optimal in the $\mathcal{P}$-x-game, but not in the $\mathcal{P}$-game. The minimax-optimal decision rule in the

$\mathcal{P}$-game is often an instance of $\mathcal{C}$-conditioning, a generalization of conditioning. The Monty Hall problem showed, however, that this is not always the case. On the other hand, if instead of the minimax criterion, we insist that update rules are calibrated, then $\mathcal{C}$-conditioning is always the right thing to do after all.

There are two more senses in which conditioning is the right thing to do. First, Walley [1991] shows that, in a sense, conditioning is the only updating rule that is *coherent*, according to his notion of coherence. He justifies coherence decision theoretically, but not by using the minimax criterion. Note that the minimax criterion puts a total order on decision rules. That is, we can say that δ is at least as good as δ' if

$$\max_{\Pr \in \mathcal{P}} E_{\Pr}[L_\delta] \leq \max_{\Pr \in \mathcal{P}} E_{\Pr}[L_{\delta'}].$$

By way of contrast, Walley [1991] puts a partial order on decision rules by taking δ to be at least as good as δ' if

$$\max_{\Pr \in \mathcal{P}} E_{\Pr}[L_\delta - L_{\delta'}] \leq 0.$$

Since both $\max_{\Pr \in \mathcal{P}} E_{\Pr}[L_\delta - L_{\delta'}]$ and $\max_{\Pr \in \mathcal{P}} E_{\Pr}[L_{\delta'} - L_\delta]$ may be positive, this is indeed a partial order. If we use this ordering to determine the optimal decision rule then, as Walley shows, conditioning is the only right thing to do.

Second, in this paper, we interpreted "conditioning" as conditioning the full given set of distributions $\mathcal{P}$. Then conditioning is not always an a priori minimax optimal strategy on the observation $X = x$. Alternatively, we could first somehow select a *single* $\Pr \in \mathcal{P}$, condition $\Pr$ on the observed $X = x$, and then take the optimal action relative to $\Pr \mid X = x$. It follows from Theorem 3.1 that the minimax-optimal decision rule δ^* in a $\mathcal{P}$-game can be understood this way. It defines the optimal response to the distribution $\Pr^* \in \Delta(\mathcal{X} \times \mathcal{Y})$ defined in Theorem 3.1(b)(ii). If $\mathcal{P}$ is convex, then $\Pr^* \in \mathcal{P}$. In this sense, the minimax-optimal decision rule can always be viewed as an instance of "conditioning," but on a single special $\Pr^*$ that depends on the loss function L rather than on the full set $\mathcal{P}$.

It is worth noting that Grove and Halpern [1998] give an axiomatic characterization of conditioning sets of probabilities, based on axioms given by van Fraassen [1987, 1985] that characterizing conditioning in the case that uncertainty is characterized by a single probability measure. As Grove and Halpern point out, their axioms are not as compelling as those of van Fraassen. It would be interesting to know whether a similar axiomatization can be used to characterize the update notions that we have considered here.

Acknowledgments Peter Grünwald was supported by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. Joseph Halpern was supported in part by NSF under grants ITR-0325453 and IIS-0534064, and by AFOSR under grant FA9550-05-1-0055.

References

Augustin, T. (2003). On the suboptimality of the generalized Bayes rule and robust Bayesian procedures from the decision theoretic point of view. In *Proceedings ISIPTA '03*, pp. 31–45.

Cozman, F. G. and P. Walley (2001). Graphoid properties of epistemic irrelevance and independence. In *Proceedings ISIPTA '01*, pp. 112–121.

Dawid, A. (1982). The well-calibrated Bayesian. *Journal of the American Statistical Association 77*.

Gärdenfors, P. and N. Sahlin (1982). Unreliable probabilities, risk taking, and decision making. *Synthese 53*, 361–386.

Gilboa, I. and D. Schmeidler (1989). Maxmin expected utility with a non-unique prior. *Journal of Mathematical Economics 18*, 141–153.

Grove, A. J. and J. Y. Halpern (1998). Updating sets of probabilities. In *Proceedings UAI '98*, pp. 173–182.

Grünwald, P. and A. Dawid (2004). Game theory, maximum entropy, minimum discrepancy, and robust Bayesian decision theory. *The Annals of Statistics 32*(4), 1367–1433.

Grünwald, P. and J. Halpern (2004). When ignorance is bliss. In *Proceedings UAI '04*, pp. 226–1.734.

Grünwald, P. and J. Halpern (2007). A game-theoretic analysis of updating sets of probabilities. http://arxiv.org/abs/0711.3235.

Herron, T., T. Seidenfeld, and L. Wasserman (1997). Divisive conditioning: Further results on dilation. *Philosophy of Science 64*, 411–444.

Hughes, R. I. G. and B. C. van Fraassen (1985). Symmetry arguments in probability kinematics. In P. Kitcher and P. Asquith (Eds.), *PSA 1984*, Volume 2, pp. 851–869. Philosophy of Science Association.

Mosteller, F. (1965). *Fifty Challenging Problems in Probability with Solutions*. Addison-Wesley.

Seidenfeld, T. (2004). A contrast between two decision rules for use with (convex) sets of probabilities: γ-maximin versus E-admissibility. *Synthese*.

Seidenfeld, T. and L. Wasserman (1993). Dilation for convex sets of probabilities. *Annals of Statistics 21*, 1139–1154.

van Fraassen, B. C. (1987). Symmetries of personal probability kinematics. In N. Rescher (Ed.), *Scientific Enquiry in Philsophical Perspective*, pp. 183–223. University Press of America.

vos Savant, M. (Sept. 9, 1990). Ask Marilyn. *Parade Magazine*, 15.

Vovk, V., A. Gammerman, and G. Shafer (2005). *Algorithmic Learning in a Random World*. Springer.

Wald, A. (1950). *Statistical Decision Functions*. Wiley.

Walley, P. (1991). *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall.

Sampling First Order Logical Particles

Hannaneh Hajishirzi and Eyal Amir
Department of Computer Science
University of Illinois at Urbana-Champaign
{hajishir, eyal}@uiuc.edu

Abstract

Approximate inference in dynamic systems is the problem of estimating the state of the system given a sequence of actions and partial observations. High precision estimation is fundamental in many applications like diagnosis, natural language processing, tracking, planning, and robotics. In this paper we present an algorithm that samples possible deterministic executions of a probabilistic sequence. The algorithm takes advantage of a compact representation (using first order logic) for actions and world states to improve the precision of its estimation. Theoretical and empirical results show that the algorithm's expected error is smaller than propositional sampling and Sequential Monte Carlo (SMC) sampling techniques.

1 Introduction

Important AI applications like natural language processing, program verification, tracking, and robotics involve stochastic dynamic systems. The current state of such systems is changed by executing actions. Reasoning is the task of computing the posterior probability over the state of such dynamic systems given past actions and observations. Reasoning is difficult because the system's exact initial state or the effects of its actions are uncertain (e.g., there may be some noise in the system or its actions may fail).

Exact reasoning (e.g., [11; 1]) is not tractable for long sequence of actions in complex systems. This is because domain features become correlated after some steps, even if the domain has much conditional-independence structure [4]. Therefore, approximate reasoning is of much interest. One of the most commonly used classes of techniques for approximate reasoning is SMC sampling [5]. These methods are efficient, but they require many samples (exponential in the dimensionality of the domain) to yield a lower error rate. Recently, [8] introduced a new sampling approach which achieves higher precision than SMC techniques given a fixed number of samples. Still, it requires a large number of samples in complex domains because in their method the domains are represented using propositional logic.

In this paper we present a sampling-based filtering algorithm in a first order dynamic system called *Probabilistic Relational Action Model (PRAM)*. We show that our new algorithm takes fewer samples and yields better accuracy than previous sampling techniques. Such improvement is possible because of the underlying deterministic structure of the transition system, the compact representation of the domains using first order logic (FOL), and efficient subroutines for first order logical *regression* (e.g., [17]) and first order logical *filtering* [19; 14].

We model a PRAM (Section 2) using probabilistic situation calculus [17], extended with a first order probabilistic prior that combines FOL and probabilities in a single framework (e.g., [18; 3; 12; 15; 7; 20]). Transitions in a PRAM are modeled with probability distributions over possible deterministic executions of probabilistic actions (every transition model can be represented this way).

Our algorithm (Section 3) first samples sequences of deterministic actions, called *first order (FO) particles*, that are possible executions of the given probabilistic action sequence and are consistent with the observations. It also updates the current state of the system given each FO particle. Next, the algorithm computes the probability of the query given the updated current state. To do so, it applies logical regression to the query for each FO particle, computes a FOL formula that represents all the possible initial states, and computes the prior probability of that formula. Finally, the algorithm computes the posterior probability as the weighted sum of these derived prior probabilities.

This algorithm achieves superior precision with fewer samples than SMC sampling techniques [5]. The intuition behind this improvement is that each FO particle corresponds to exponentially many state sequences (particles) generated

by earlier techniques. The algorithm is computationally efficient when FOL regression and filtering with the deterministic actions are efficient. We prove our claim about precision and verify the results empirically by several experiments (Section 4).

Our representation for PRAM differs from Dynamic Bayesian Networks (DBNs) (e.g., [13]). DBNs emphasize conditional independence among random variables. In contrast, our model applies a representation for the transition model as a distribution over deterministic actions. Also, PRAM uses a different model for observations. The observations are received asynchronously without prediction of what will be observed. Both frameworks are universal and can represent each other, but they are more compact and natural in different scenarios.

The closest work to ours is [8] which also samples deterministic executions of the given probabilistic sequence. However, that work assumes that the deterministic sequences are always executable. In this paper we relax this assumption and avoid sampling the sequences that are not executable by keeping track of the current state of the system. Also, our representation is more compact (uses FOL), so it achieves higher precision with less samples.

Recently, [10; 22] explore reasoning algorithms in relational HMMs. They use a different representation than ours for their observations. Moreover, they do not use a compact representation for their prior distribution. Earlier algorithm trades efficiency of computation for precision as it is an exact method. Latter, introduced a sampling algorithm where each sample represents a set of states and is generated from the probability distribution over disjoint sets of the states. Therefore, to update this distribution they build new disjoint sets at each time step which leads to a large number of disjoint sets (exponential in domain features) in long sequences. Our algorithm is different in a sense that we do not sample states; we sample deterministic sequences which correspond to state sequences that are derived by regressing the query through the deterministic sequence.

2 Probabilistic Relational Action Models

In this section we present our framework, called *Probabilistic Relational Action Model* (PRAM), for representing the dynamic system. The system is dynamic in a sense that its state changes by executing actions. Actions have probabilistic effects that are represented with a probability distribution over possible deterministic executions.

A PRAM consists of two main parts: (1) A prior knowledge representing a probability distribution P^0 over initial world states in a relational model. (2) A probability distribution *PA* over deterministic executions of probabilistic actions. In what follows we define the basic building blocks of a PRAM. We first define the language $\mathcal{L}$ of a PRAM for representing the probability distributions:

Definition 1. [PRAM language] The language $\mathcal{L}$ of a PRAM is a tuple $(F, C, V, \mathcal{A}, D\mathcal{A})$ consisting of:

- F a finite set of predicate variables (called *fluents*) whose values change over time.
- C a finite set of constants representing objects in the domain.
- V a finite set of variables.
- $\mathcal{A}, D\mathcal{A}$ finite sets of probabilistic and deterministic action names, respectively.

We define a *fluent atom* as a formula of the form $f(x_1, \ldots, x_k)$ (also represented by $f(\vec{x})$), where $x_1, \ldots, x_k \in V \cup C$ are either variables or constants. In PRAM grounding of a fluent $f(\vec{x})$ is defined as replacing each variable in $\vec{x}$ with a constant $c \in C$. Accordingly, a world state $s \in \mathcal{S}$ is defined as a full assignment of $\{true, false\}$ to all the groundings of all the fluents in F.

Example 1 (Briefcase)**.** *Briefcase* is a domain consisting of *objects*, *locations*, and a *briefcase* B. The variables $?o$ and $?l$ denote *objects* and *locations*, respectively. The fluents are: $\mathit{In}(?\mathrm{o}), \mathit{At}(B, ?l), \mathit{At}(?o, ?l)$. An agent interacts with the system by executing some probabilistic actions: *putting* objects in, $\mathit{PutIn}(?o, B)$, *taking* objects *out* of the briefcase, $\mathit{TakeOut}(?o, B)$, and *moving* the briefcase with objects inside, $\mathit{Move}(B, ?l_1, ?l_2)$. Some deterministic actions in the domain are: *MvWithObj*, *PutInSucc*, and *PutInFail*.

In PRAM each deterministic action $da(\vec{x}) \in D\mathcal{A}$ is specified by *precondition* and *successor state* axioms [17].

Definition 2. [Deterministic action axioms] Precondition axioms show the conditions under which the deterministic action da is executable in a given state; *Precond* is a special predicate denoting the executability of a deterministic action: $\mathit{Precond}_{da}(\vec{x}) \Leftrightarrow \Phi(\vec{x})$ where Φ is a FOL formula.

Successor state axioms enumerate all the ways that the value of a particular fluent can be changed; A successor state axiom for a fluent f is defined as:

$$\mathit{Precond}^t_{da}(\vec{x}) \Rightarrow (\mathit{Succ}^t_{f,da}(\vec{x}) \Leftrightarrow f^{t+1}(\vec{x}))$$

where $\mathit{Succ}^t_{f,da}(\vec{x})$ is a FOL formula at time t. One can easily derive the effects of actions by the above axioms.

For example, the precondition of the deterministic action $\mathit{MvWithObj}(B, ?l_1, ?l_2)$ is $\mathit{At}(B, ?l_1) \wedge \neg \mathit{At}(B, ?l_2)$, and its effect is $\neg \mathit{At}(B, ?l_1) \wedge \mathit{At}(B, ?l_2) \wedge (\forall ?o\ \mathit{In}(?o) \Rightarrow \mathit{At}(?o, ?l_2))$.

The grounding of a deterministic action is defined as the grounding of all the fluent atoms appearing in the precondition and effect logical formulas $\mathit{Precond}^t_{da}(\vec{x})$ and $\mathit{Succ}^t_{f,da}(\vec{x})$. Therefore, a grounded deterministic action is a transition function $\mathcal{T} : \mathcal{S} \times D\mathcal{A} \rightarrow \mathcal{S}$. One can see from the example that for grounding the action *MvWithObj*, one

needs to permute all the possible combinations of *objects* inside the *briefcase*. This increases the dimensionality of the domain and increases the required number of samples to yield low error. Hereinafter for simplicity, we represent fluents and actions without their arguments whenever it is not necessary to mention the variables or domain objects.

Definition 3. [Probability distribution for probabilistic actions] Let $\psi_1 \dots \psi_k$ be FOL formulas (called *partitions*) that divide the world states into mutually disjoint sets. Then, $PA(da|a,s)$ is defined as a probability distribution over possible deterministic executions da of the probabilistic action a in the state s which satisfies one of the partitions ψ_i $(i \leq k)$. More formally, when some state s satisfies partition ψ_i then $PA(da|a,s) = PA_i(da)$, where PA_i is a probability distribution over different deterministic executions da of action a corresponding to the partition ψ_i.

$$PA(da|a,s) = \begin{cases} PA_1(da) & s \models \psi_1 \\ PA_2(da) & s \models \psi_2 \\ \dots \end{cases} \quad (1)$$

We assume that replacing variables in $a(\vec{x})$ with constants does not change $PA(da(\vec{x})|a(\vec{x}),s)$.

For example, for probabilistic action *TakeOut*$(?o)$, deterministic action *TakeOutSucc*$(?o)$, and $s \models \psi_1 = In(?o)$: $PA(da|a,s) = PA_1(TakeOutSucc(?o)) = 0.9$.

In PRAM we assume a prior distribution over world states at time 0. Models like [18] are used to represent probabilities in relational models. A knowledge engineer can define the semantics of the prior distribution by using each of these models. Then, we use the inference algorithm defined in that model's semantics to compute probability of formulas at time 0. Our filtering algorithm does not depend on the way the initial knowledge has been represented. We discuss more about this in Section 3.1.3.

In conclusion, we define a PRAM formally as follows:

Definition 4. A PRAM is a tuple $(\mathcal{L}, AX, PA, P_0)$ as:

- Language $\mathcal{L} = (F, C, V, \mathcal{A}, D\mathcal{A})$ representing the language of PRAM (Definition 1)
- A set of deterministic action axioms *AX* (Definition 2)
- A probability distribution *PA* for each probabilistic action (Definition 3)
- A prior distribution P^0 over initial world states

Based on the aforementioned dynamic system, PRAM, we define our stochastic filtering problem as computing probability of a query given a sequence of probabilistic actions and observations. The query is defined as a FOL formula φ^T at the final time step, T. Note that throughout the paper superscripts represent time. Each a^t in the probabilistic action sequence $\langle a^1, \dots, a^T \rangle$ represents the probabilistic action that has been executed at time t.

The observations $\langle o^0, \dots, o^T \rangle$ are given asynchronously in time without prediction of what we will observe (thus, this is different from HMMs [16], where a sensor model is given). Each observation o^t is represented with a FOL formula over fluents. When o^t is observed at time t, the FOL formula o^t is true about the state of the world at time t. We assume that a deterministic execution da of the probabilistic action a^t in the sequence does not depend on the future observations.

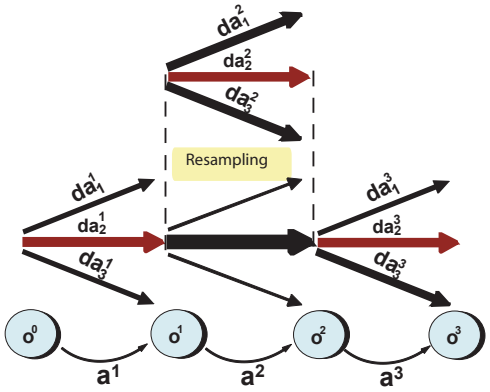


Figure 1: Sampling the FO particle $\langle da_2^1, da_2^2, da_2^3 \rangle$ (straight arrows) given probabilistic sequence $\langle a^1, a^2, a^3 \rangle$ (curved arrows) and observations $\langle o^0, o^1, o^2, o^3 \rangle$. Each deterministic action da^t is sampled from distribution $P(da^t|a^t, da^{1:t-1}, o^{0:t})$. The thickness of the arrows represent the weight of the current FO particle.

3 Filtering Algorithm

In this section we present our filtering algorithm for computing probability of a query φ^T given a sequence of probabilistic actions $a^{1:T} = \langle a^1, \dots, a^T \rangle$ and observations $o^{0:T} = \langle o^0, \dots, o^T \rangle$ in a PRAM. The algorithm approximates this probability by generating samples among possible deterministic executions of the given probabilistic action sequence. Then, it places those samples instead of the enumeration of deterministic executions and marginalizes over those samples. The following equation shows the exact computation.

$$P(\varphi^T|a^{1:T}, o^{0:T}) = \sum_i P(\varphi^T|\vec{DA}_i, o^{0:T}) P(\vec{DA}_i|a^{1:T}, o^{0:T}) \quad (2)$$

where $\vec{DA}_i$ is a possible execution of the sequence $a^{1:T}$.

The first step of our approximate algorithm is generating N samples (called *FO particles*) from all the possible deterministic executions of the given probabilistic sequence. Two different sampling algorithms are introduced in Section 3.2. The algorithms (illustrated in Figure 1) generate a weighted FO particle $\vec{DA}_i$ with weight w_i given the sequence $a^{1:T}$ and the observations $o^{0:T}$ from the probability distribution $P(\vec{DA}_i|a^{1:T}, o^{0:T})$.

The next step of the algorithm is computing $P(\varphi^T|\vec{DA}_i, o^{0:T})$ for each FO particle $\vec{DA}_i$. It does

PROCEDURE *FOFA*($SampleAlg$, φ^T, $a^{1:T}$, $o^{0:T}$)
Input: Sampling algorithm $SampleAlg$, probabilistic sequence $a^{1:T}$, observations $o^{0:T}$, and query φ^T
Output: $\tilde{P}(\varphi^T|a^{1:T}, o^{0:T})$
1. $(\vec{DA}_{1:N}, curF_{1:N}) \leftarrow SampleAlg(a^{1:T}, o^{0:T})$
2. for each $\vec{DA}_i$ compute $PFOF(\varphi^T, \vec{DA}_i, curF_i)$
3. **return** $\tilde{P}(\varphi^T|a^{1:T}, o^{0:T})$ using Equation (6).

Figure 2: FOFA: First Order Filtering Algorithm for approximating $P(\varphi^t|a^{1:t}, o^{0:t})$ given a sampling algorithm *SampleAlg* (either *S/R-Actions* (Figure 5) or *S-Actions* (Figure 6)). *Italic fonts* is used to denote subroutines.

so (Section 3.1) by updating the current state of the system and then computing the posterior probability conditioning on the updated current state.

Finally, the algorithm uses generated samples in place of $\vec{DA}_i$ in Equation (2) and computes $\tilde{P}_N(\varphi^T|a^{1:T}, o^{0:T})$ as an approximation for the posterior probability of the query φ^T given the sequence $a^{1:T}$ and the observations $o^{0:T}$ by using the Monte Carlo integration [5]:

$$\tilde{P}_N(\varphi^T|a^{1:T}, o^{0:T}) = \sum_i w_i P(\varphi^T|\vec{DA}_i, o^{0:T}) \qquad (3)$$

Details of each step of our first order filtering algorithm (*FOFA*, Figure 2) are explained next. We first present the step of computing $P(\varphi^T|\vec{DA}_i, o^{0:T})$ because it is used as a subroutine in the sampling algorithms.

3.1 Probability of a FOL Formula at time t

In this section we show how to compute the probability $P(\varphi^t|\vec{DA}, o^{0:t})$ of the formula φ^t given a FO particle $\vec{DA}$ and observations $o^{0:t}$. Executing the FO particle $\vec{DA}$ with observations $o^{0:t}$ updates the *current state* of the system. The current state is derived by applying a FO *progression* subroutine (described below) at each time step. Afterwards, procedure *PFOF* (Figure 3) computes the probability of the query given the current state of the system for each FO particle. Its first step applies a FO *regression* subroutine to the query and the current state formula and as output returns FOL formulas at time 0. This can be done since the actions are deterministic. The algorithm's second step computes the prior probability of the regression of the query conditioned on the current state formula regressed by the FO particle; Recall that a FO particle is a sampled sequence of deterministic actions.

3.1.1 Progress Current State Formula

We define the current state formula as a FOL formula representing the set of states that are true after executing a sequence of deterministic actions and receiving observations. In this section we present algorithm *Progress* that updates the current state formula given a deterministic action and an observation. In general, progressing a FOL formula δ with a deterministic action da, $Progress(\delta, da)$, results in a set of FOL formulas $\Delta(p_{1:n})$ where $p_1, \ldots, p_n$ are atomic subformulas of Δ and $\delta \wedge Precond_{da} \models \Delta(Succ_{p_1,da}, \ldots, Succ_{p_n,da})$ (see [19] for more details). Also, filtering a FOL formula δ with an observation o is the FOL formula $\delta \wedge o$.

Overall, generating all Δs is impossible because there are infinitely many such Δs. In this paper we assume that progression of the current state formula *curF* with a deterministic action da, $Progress(curF, da, o)$, is representable with a FOL formula. Hence, it is equal to $\bigvee_i \Delta_i(p_{1:n}) \wedge o$. Furthermore, [19; 14] provide some special conditions that the progression algorithm is polynomial and the representation of progressing a formula is compact. For example, progressing $In(O)$ through deterministic action $MvWithObj(L1, L2)$ results in the formula $At(B, L2) \wedge In(O) \wedge At(O, L2) \wedge (\forall ?o\ In(?o) \Rightarrow At(?o, L2))$.

PROCEDURE *PFOF*(φ^t, $\vec{DA}$, *curF*)
Input: FO particle $\vec{DA}$ and the current state formula *curF*
Output: $P(\varphi^t|\vec{DA}, o^{0:t})$
1. $\varphi^0 \leftarrow RegSeq(\varphi^t, \vec{DA})$
2. $curF^0 \leftarrow RegSeq(curF, \vec{DA})$
3. $p_0 \leftarrow Prior\text{-}FOF(\varphi^0|curF^0)$
4. **return** $P(\varphi^t|\vec{DA}, o^{0:t}) \leftarrow p_0$

PROCEDURE *Prior-FOF*(φ^0, M_{MLN})
Input: Formula φ^0, Markov network M_{MLN} of the prior MLN
Output: $P(\varphi^0)$
1. $\bigwedge_i C_i \leftarrow$ ConvertToCNF(φ^0)
2. Define indicator functions as Equation (4)
3. $M \leftarrow ConstructNetwork(M_{MLN}, fluents\ of\ \varphi^0)$
4. **return** $P(\varphi^0)$ by performing inference on M

Figure 3: *PFOF*: Compute probability $P(\varphi^t|\vec{DA}, o^{0:t})$ of a FOL formula φ^t given a FO particle $\vec{DA}$ and the current state formula.

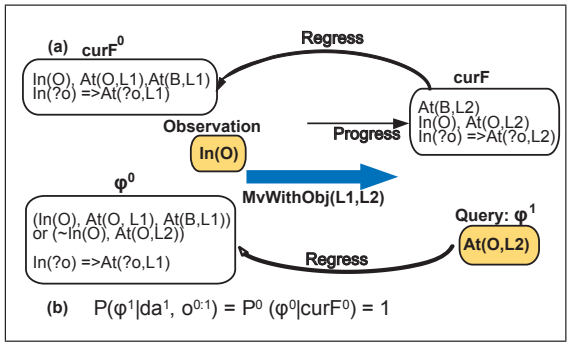


Figure 4: (a) Regressing query φ^1 and *curF* by *MvWithObj*. Note, $L1$, $L2$, and O are constants, and there are only two possible locations $L1$ and $L2$, i.e. $At(O, L2) \equiv \neg At(O, L1)$. (b) Computing $P(\varphi^2|\vec{DA}, o^{0:1})$ using Lemma 1.

3.1.2 Regressing a FOL Formula

Procedure *RegSeq* takes a FOL formula φ^t and a FO particle $\vec{DA}$ and returns as output another FOL formula φ^0. φ^0 represents the set of possible initial states, given that the fi-

nal state satisfies φ^t, and the FO particle $\vec{DA}$ occurs. Thus, every state that satisfies φ^0 leads to a state satisfying φ^t after $\vec{DA}$ occurs.

For a deterministic action da^t and a FOL formula φ^t, the regression $\mathit{Regress}(\varphi^t, da^t)$ of φ^t through da^t is a FO formula φ^{t-1} such that state s^{t-1} satisfies φ^{t-1} iff the result of the transition function $\mathcal{T}(s^{t-1}, da^t)$ satisfies φ^t. The computation of the regression $\mathit{RegSeq}(\varphi^t, \vec{DA})$ of φ^t through the FO particle $\vec{DA}$ is done recursively.

$$\mathit{RegSeq}(\varphi^t, \langle da^1, ..., da^t\rangle) = \mathit{RegSeq}(\mathit{Regress}(\varphi^t, da^t), \langle da^1, ..., da^{t-1}\rangle).$$

Regression of the current state formula *curF* is defined similar to the regression of formula φ^t since the current state is also represented with a fist order formula.

The algorithm for regression $\mathit{Regress}(\varphi^t, da)$ of formula φ^t with deterministic action da works as follows (see [17]). Suppose that φ^t is an atomic fluent $f^t(\vec{x})$ with the successor state axiom $\mathit{Succ}^{t-1}_{f,da}(\vec{x})$. Then, we derive the regression of φ^t by replacing $f^t(\vec{x})$ with $\mathit{Succ}^{t-1}_{f,da}(\vec{x})$. The regression of non-atomic formulas are derived inductively as follows:

- $\mathit{Regress}(\neg\varphi^t) = \neg\mathit{Regress}(\varphi^t)$
- $\mathit{Regress}(\varphi^t_1 \wedge \varphi^t_2) = \mathit{Regress}(\varphi^t) \wedge \mathit{Regress}(\varphi^t_2)$
- $\mathit{Regress}((\exists\nu)\varphi^t) = \exists\nu\mathit{Regress}(\varphi^t)$

The efficiency of the regression yields from the fact that we regress non-grounded fluents $f(\vec{x})$. Besides, one can employ some approximations at each step (like removing unnecessary clauses) to maintain the compactness of the regressed formulas.

We now summarize and show how to compute the probability distribution $P(\varphi^t|\vec{DA}, o^{0:t})$ by applying progression and regression. The algorithm first computes *curF* by progressing through the FO particle and observations. Then, it computes φ^0 and curF^0 by regressing φ^t and *curF*. Finally, it computes $P^0(\varphi^0|\mathit{curF}^0)$ as the evaluation of $P(\varphi^t|\vec{DA}, \mathit{curF})$ as shown by the following lemma.

Lemma 1. Let φ^t be the query and $o^{0:t}$ the observations. If *curF* is the current state formula, $\varphi^0 = \mathit{RegSeq}(\varphi^t, \vec{DA})$, and $\mathit{curF}^0 = \mathit{RegSeq}(\mathit{curF}, \vec{DA})$, then $P(\varphi^t|\vec{DA}, o^{0:t}) = P^0(\varphi^0|\mathit{curF}^0)$.

Proof. Probability of a formula is a marginalization over states: $P(\varphi^t, o^{0:t}|\vec{DA}) = \sum_s P(\varphi^t, \mathit{curF}|s, \vec{DA})P(s|\vec{DA})$. For s, $P(\varphi^t, \mathit{curF}|s, \vec{DA}) = 1$ if $s \models \varphi^0 \wedge curF^0$, o.w. it is 0. The reason is that executing the deterministic sequence $\vec{DA}$ in state s results at time t that models φ^t and is consistent with observations $o^{0:t}$ iff $s \models \varphi^0 \wedge curF^0$. Therefore, $P(\varphi^t, o^{0:t}|\vec{DA}) = \sum_{s\models\varphi^0, \mathit{curF}^0} P(s) = P^0(\varphi^0, \mathit{curF}^0)$. The same computations exist for $P(o^{0:t}|\vec{DA})$. Therefore,

$$P(\varphi^t|\vec{DA}, o^{0:t}) = \frac{P(\varphi^t, o^{0:t}|\vec{DA})}{P(o^{0:t}|\vec{DA})} = \frac{P^0(\varphi^0, \mathit{curF}^0)}{P^0(\mathit{curF}^0)}. \quad \blacksquare$$

Figure 4 shows an example of computing $P(\varphi^t|\vec{DA}, o^{0:t})$ using procedure *PFOF*. The next section shows how to compute $P^0(\varphi^0|\mathit{curF}^0)$ using the prior P^0. Note that φ^0 and curF^0 are FOL formulas over fluents at time 0.

3.1.3 Prior Probability of a FOL Formula

In this section we describe procedure *Prior-FOF* to compute the probability of a FOL formula at time 0. We assume that the prior distribution P^0 over world states of a PRAM with language $\mathcal{L} = (F, C, V, \mathcal{A}, D\mathcal{A})$ is represented by a Markov Logic Network (MLN) [18]. We choose to represent our prior probabilistic logic with an MLN (called *prior MLN*) because it keeps the expressive power of FOL for a fixed domain.

Our prior MLN consists of a set of weighted FOL formulas over the fluents in language $\mathcal{L}$ of the PRAM. The semantics of the MLN is that of a Markov network M_{MLN} [9] whose cliques correspond to groundings of the formulas given the universe of objects $C \in \mathcal{L}$. The potential Φ of a clique Cl is defined as the exponential of the weight of the corresponding formula in case the grounding is true.

We introduce *Prior-FOF* (Figure 3) to compute the probability of a FOL formula φ^0 by slightly changing the inference algorithm expressed for MLNs. Procedure *Prior-FOF* first converts φ^0 to a clausal form; Note that like MLNs existentially quantified formulas are replaced by disjunction of their groundings. Then, *Prior-FOF* assigns an indicator function, $\mathcal{I}_{g(C_i)}$, to every grounding g of a clause C_i.

$$\mathcal{I}_{g(C_i)}(\vec{f}_{g(C_i)}) = \begin{cases} 1 & \vec{f}_{g(C_i)} \models g(C_i) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The next step is to construct a Markov network M as the minimal subset of the original network M_{MLN} required to compute $P^0(\varphi^0)$. The nodes in M consist of all the ground fluents $f_{g(C_i)}$ in clauses C_i of φ^0 and all the nodes in the original Markov network M_{MLN} with a path to the fluent $f_{g(C_i)}$. The final step is to perform inference on this network. It can be computed exactly by performing variable elimination (e.g., [9]) in $P^0(\varphi^0) \propto \sum_{f\in M} \prod_{\vec{f}_j\in Cl_j, \vec{f}_i\in C_i} \Phi_j(\vec{f}_j)\mathcal{I}(\vec{f}_i)$. Also, one can employ any approximate inference algorithm like Gibbs sampling.

We are not restricted to use MLNs as a framework for representing prior probability distribution. We can use any other probabilistic logical frameworks ([3; 12; 15; 7; 20]) provided that the expressed inference algorithm in that framework can compute probability of the query. Note that the query φ^0 for that framework is derived from the regression of the original query φ^t given an FO particle. Also, our algorithms would work for unbounded domains just by using a framework (e.g., [21]) for representing prior distribution over infinite states.

PROCEDURE *S/R-Actions*($a^{1:T}$, $o^{0:T}$)
Input: Probabilistic sequence $a^{1:T}$ and observations $o^{0:T}$
Output: N FO particles $\vec{DA}_{1:N}$
1. for $n \leftarrow 1 \ldots N$: $curF_n \leftarrow o^0$
2. for $t \leftarrow 1 \ldots T$
3. for $n \leftarrow 1 \ldots N$
4. $\pi \leftarrow \sum_{\psi_i} PA_i \cdot PFOF(\psi_i^{t-1}, da^{1:t-1})$
5. $P \leftarrow \sum_{\psi_i} PA_i \cdot PFOF(\psi_i^{t-1}, da_n^{1:t-1}, curF_n)$
6. $\hat{da^t} \leftarrow$ a sample from probability distribution π
7. $curF_n \leftarrow Progress(curF_n, \hat{da^t}_n, o^t)$
8. $w_n^* \leftarrow \frac{\pi(\hat{da_n^t})}{P(\hat{da_n^t})}$
9. $\langle \hat{w}_1, \ldots, \hat{w}_N \rangle \leftarrow Normalize(w_1^*, \ldots, w_N^*)$
10. $(da_{1:N}^t, w_{1:N}) \leftarrow Resample(\hat{da^t}_{1:N}, \hat{w}_{1:N})$
11. **return** $\vec{DA}_{1:N} \leftarrow \langle da^1 \ldots da^T \rangle_{1:N}$

Figure 5: *S/R-Actions*: Sampling/Resampling algorithm for generating N FO particles given a sequence $a^{1:T}$ and observations $o^{0:T}$.

PROCEDURE *S-Actions*($a^{1:T}$, $o^{0:T}$)
Input: Probabilistic sequence $a^{1:T}$ and observations $o^{0:T}$
Output: N FO particles $\vec{DA}_{1:N}$
1. for $n \leftarrow 1 \ldots N$: $curF_n \leftarrow o^0$
2. for $t \leftarrow 1 \ldots T$
3. for $n \leftarrow 1 \ldots N$
4. $P \leftarrow \sum_{\psi_i} PA_i \cdot PFOF(\psi_i^{t-1}, da^{1:t-1}, curF_n)$
5. $da^t \leftarrow$ a sample from probability distribution P
6. $curF_n \leftarrow Progress(curF_n, da^t, o^t)$
7. **return** $\vec{DA}_{1:N} \leftarrow \langle da^1 \ldots da^T \rangle_{1:N}$

Figure 6: *S-Actions*: Direct sampling algorithm for generating N FO particles given a sequence $a^{1:T}$ and observations $o^{0:T}$.

3.2 Sampling Algorithms

In this section we describe procedures *S/R-Actions* (Figure 5) and *S-Actions* (Figure 6) which generate N samples (called *FO particles*) given a sequence of probabilistic actions and observations. Each FO particle is a possible deterministic execution of the given probabilistic sequence. Both algorithms incrementally build every FO particle by sampling a deterministic action at a time. Later, we will discuss about the deficiencies of *S/R-Actions* algorithm.

Both *S/R-Actions* and *S/Actions* generate a FO particle $\vec{DA} = \langle da^1, \ldots, da^T \rangle$ given a sequence $a^{1:T}$ and observations $o^{0:T}$ from the distribution $P(\vec{DA}|a^{1:T}, o^{0:T})$. We compute this probability distribution iteratively:

$$P(\vec{DA}|a^{1:T}, o^{0:T}) = P(da^1|a^1, o^{0:1}) \prod_t P(da^t|a^t, da^{1:t-1}, o^{0:t})$$

The above derivation allows iterative sampling of deterministic actions from the distribution $P(da^t|a^t, da^{1:t-1}, o^{0:t})$. Recall that a FO particle is a sequence of deterministic actions. Thus, at each time the algorithms sample a deterministic action da^t given the probabilistic action a^t, the previous deterministic actions $da^{1:t-1}$, and the previous observations $o^{0:t}$; Note that current deterministic action is independent of the future observations. Then, the algorithms update the current state formula *curF* (Section 3.1.1) given the current deterministic action and the observation.

Algorithms *S/R-Actions*, *S-Actions* use different approaches for incremental sampling of each deterministic action. In *S/R-Actions*, we adopt a sequential importance sampling approach. At each time step, *S/R-Actions* samples deterministic actions based on a normalized importance function, assigns weights to the particles, and resamples if the weights have high variance. The importance function $\pi(da^t|a^t, da^{1:t-1})$ approximates $P(da^t|a^t, da^{1:t-1}, o^{0:t})$ by ignoring the effect of the current state *curF* in the $PRAM = (\mathcal{L}, AX, PA, P^0)$.

$$\pi(da^t|a^t, da^{1:t-1}) = \sum_i PA_i(da^t) PFOF(\psi_{i,a^t}^{t-1}, da^{1:t-1})$$

where ψ_{i,a^t} is the i^{th} partition of action a^t (Definition 3).

At time step t, *S/R-Actions* samples N deterministic actions $da_{1:N}^t$ from the importance function: $\pi(da^t|a^t, da^{1:t-1})$. It then restructures the n^{th} particle $\vec{DA}_n^{t-1}$ by attaching the deterministic action da_n^t to that particle, i.e. $\vec{DA}_n^t = \langle da_n^1, \ldots, da_n^{t-1}, da_n^t \rangle$. The importance weight w_n^* of the n^{th} particle is derived as: $w_n^* = \frac{P(da_n^t|a^t, da_n^{1:t-1}, o^{0:t})}{\pi(da_n^t|a^t, da_n^{1:t-1})}$.

Accordingly, the normalized importance weight w_n of the n^{th} particle is: $w_n = \frac{w_n^*}{\sum_{i=1}^N w_i^*}$.

The exact value for $P(da^t|a^t, da^{1:t-1}, o^{0:t})$ is computed using *PFOF* subroutine (Figure 3).

$$P(da^t|a^t, da^{1:t-1}, o^{0:t}) = \sum_i PA_i(da^t) \cdot PFOF(\psi_{i,a^t}^{t-1}, da^{1:t-1}, curF) \quad (5)$$

S/R-Actions resamples the particles if the variance of the normalized importance weights becomes too high. The basic idea of resampling is to avoid those particles with very low weight and concentrate on particles that have higher normalized weight. An estimation (see [5]) for measuring high variance among the weights is estimating the *effective number* of particles as $\widehat{N_{eff}} = \frac{1}{\sum_{i=1}^N w_i}$. If $\widehat{N_{eff}}$ is smaller than a threshold then procedure *Resample* generates a new deterministic action da^t from the probability distribution over the normalized weights $w_{1:N}$ of the particles. It then assigns equal weights, $\frac{1}{N}$, to all the particles.

The second sampling algorithm, *S-Actions* (Figure 6), is derived by simplifying the above sampling algorithm. This algorithm at each time step generates samples among executable deterministic actions. It samples every deterministic action in a FO particle from the exact computation for $P(da^t|a^t, da^{1:t-1}, o^{0:t-1})$ (Equation 5) instead of sampling from the importance function and assigning weights. Therefore, all the weights are equal to $\frac{1}{N}$.

S/R-Actions, samples deterministic actions even when they are not executable and assigns weight zero to those particles. Therefore, it decreases the effective number of samples. Resampling step does not help that much because it resamples the latest deterministic action in the FO particle. However, resampling earlier deterministic actions may result in a more effective set of FO particles. Note that this new type of resampling is not tractable. Therefore, it makes more sense to maintain the current state formula *curF* as in *S-Actions* and just sample executable deterministic actions (as in *S-Actions*) unless there exists a better resampling procedure. In the empirical results we examine the effects of using *S/R-Actions* and *S-Actions* sampling algorithms in the *FOFA* filtering algorithm (Figure 2).

3.3 Correctness, Complexity, and Accuracy

The following theorem shows how *FOFA* with *S-Actions* and *S/R-Actions* compute the approximate posterior distribution $\tilde{P}_N(\varphi^T|a^{1:T}, o^{0:T})$.

Theorem 1. Let φ^T be the query, $a^{1:T}$ be the given probabilistic sequence, $\vec{DA}_i$ be the FO particles, and $o^{0:T}$ be the observations. If $curF_i$ is the current formula given the i^{th} FO particle, $\varphi_i^0 = RegSeq(\varphi^T, \vec{DA}_i)$, and $curF_i^0 = RegSeq(curF_i, \vec{DA}_i)$, then

$$\tilde{P}_N(\varphi^T|a^{1:T}, o^{0:T}) = \sum_i w_i P^0(\varphi_i^0|curF_i^0) \quad (6)$$

where $w_i = \frac{1}{N}$ for *S-Actions*. Besides, for *S-Actions*:

$$\tilde{P}_N(\varphi^T|a^{1:T}, o^{0:T}) \rightarrow_{N\rightarrow\infty} P(\varphi^T|a^{1:T}, o^{0:T}) \quad (7)$$

The proof follows from using MC integration in Equation 2 and using Lemma 1.

The running time R_{FOFA} of our filtering algorithm (Figure 2) is $O(N \cdot T \cdot (R_{RegSeq} + R_{Progress} + R_{Prior\text{-}FOF}))$, where N is the number of samples and T is the length of the given sequence of probabilistic actions. Efficiency of *FOFA* results from efficiency of the underlying algorithms for *RegSeq*, *Progress*, and *Prior-FOF*.

We evaluate the accuracy of our sampling algorithm *FOFA* by computing expected KL-distance [1] as the expected value of all the KL-distances between the exact distribution P and the approximation $\tilde{P}$ derived by *FOFA*. Our algorithm, *FOFA*, has higher accuracy than SMC for a fixed number of samples. The intuition is that each FO particle generated by *FOFA* covers many particles generated by SMC.

Theorem 2. If *FOFA*(*S-Actions*) and SMC approximate posterior distribution $P(\varphi^T|a^{1:T}, o^{0:T})$ with N samples. Then, Expected-$KL_{FOFA(S\text{-}Actions)} \leq$ Expected-KL_{SMC}.

[1] $KL(P, \tilde{P}) = \sum_x P_x log(P_x/\tilde{P}_x), KL(P, \tilde{P}) = 0$ if $P = \tilde{P}$

The proof is similar to the proof of Theorem 3.3 in [8]. Intuitively, we define a mapping f to map each set of FO particles of *S-Actions* to sets of particles of SMC. The mapping f is defined such that it covers all the possible sets of particles of SMC, and for two separate sets of particles $z_i \neq z_j$, $f(z_i) \cap f(z_j) = \emptyset$, and $Pr_{FOFA}(z) = Pr_{SMC}(f(z))$. Furthermore, we prove that $\forall y \in f(z), KL(P, \tilde{P}_1^z) \leq KL(P, \tilde{P}_2^y)$ where $\tilde{P}_1$ and $\tilde{P}_2$ are approximations returned by *FOFA* and SMC, respectively.

4 Empirical Results

We implemented our algorithm *FOFA* (Figure 2) with both *S/R-Actions* (Figure 5) and *S-Actions* (Figure 6). Our algorithms take advantage of a different structure than that available in DBNs. Hence, we focused on planning-type domains: *briefcase* and *depots* taken from International Planning Competition at AIPS-98 and AIPS-02. [2] We randomly assigned deterministic executions and a probability distribution over them for each action. For example, for action *PutIn* we considered two executions *PutInSucc* and *PutInFail* with probabilities 0.9 and 0.1. Note that DBN representations (transitions between states) for the above frameworks are not compact because the independence assumptions among the state variables are not known.

We compared the accuracy of our *FOFA*(*S/R-Actions*) and *FOFA*(*S-Actions*) with SCAI [8] and SMC algorithms. Note that we grounded the domains for running SCAI and SMC. We ran sampling algorithms (50 times) for a fixed number of samples and computed the KL-distance between their approximation and the exact posterior. We calculated the average over these derived KL-distances to approximate the expected KL-distance. SCAI assumes that deterministic actions are always executable. We include this assumption in the *depots* domain in which we compared the results with SCAI. In the *briefcase* domain, we just compared our algorithms with the SMC techniques.

Figures 7 and 8 show the expected KL-distances (in logarithmic scale) vs. number of samples in the *depots* and *briefcase*, respectively. As we expected, the average KL-distance for *FOFA*(*S-Actions*) is always the lowest. We expect to get more improvement when there are more objects in the domain (here, we just use 5 constants to be able to compute the exact distribution). For the *briefcase* domain (Figure 8) SMC has higher accuracy than *FOFA*(*S/R-Actions*) for 1000 samples. The reason is that the posterior distribution converges to the stationary distribution in this domain (4 constants, 256 states). Even grounded domain is not too big, but for cases involving longer sequences and more states we did not have the exact posterior to compare with since the implementation for the exact algorithm crashes.

[2] Also available from: `ftp://ftp.cs.yale.edu/pub/mcdermott/domains/`.

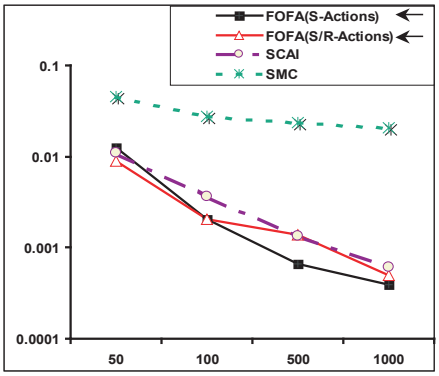


Figure 7: Expected KL-distance (in **logarithmic scale**) of our algorithms, *FOFA*(*S-Actions*) and *FOFA*(*S/R-Actions*), SCAI, and SMC with the exact distribution vs. number of samples for *depots*.

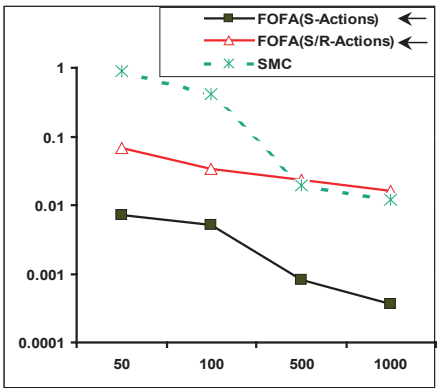


Figure 8: Expected KL-distance (in **logarithmic scale**) of our algorithms, *FOFA*(*S-Actions*) and *FOFA*(*S/R-Actions*), and SMC with the exact distribution vs. number of samples for *briefcase*.

5 Conclusions and Future Work

In this paper we presented a sampling algorithm to compute the posterior probability of a query given a sequence of actions and observations in a FO dynamic system. Our algorithm takes advantage of a compact representation and achieves higher accuracy than SMC sampling and earlier propositional sampling techniques.

There are several directions that we can continue this work: (1) Apply sampling in FO Markov Decision Processes(MDP)s [2](2) Learn the transition model in PRAM. (3) Apply the algorithm in text understanding. (4) Generalize the representation to continuous domains (e.g., by discretizing the real value variables or by combining with Rao-Blackwellised Particle Filtering [6]).

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. This work was supported by DARPA SRI 27-001253 (PLATO project), NSF CAREER 05-46663, and UIUC/NCSA AESIS 251024 grants.

References

[1] F. Bacchus, J. Y. Halpern, and H. J. Levesque. Reasoning about noisy sensors and effectors in the situation calculus. *AI*, 1999.

[2] C. Boutilier, R. Reiter, and B. Price. Symbolic dynamic programming for first-order MDPs. In *IJCAI*, 2001.

[3] V. S. Costa, D. Page, M. Qazi, and J. Cussens. Clp(bn): Constraint logic programming for probabilistic knowledge. In *UAI*, 2003.

[4] T. Dean and K. Kanazawa. Probabilistic temporal reasoning. In *AAAI*, 1988.

[5] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 1st edition, 2001.

[6] A. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *UAI*, 2000.

[7] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *IJCAI*, 1999.

[8] H. Hajishirzi and E. Amir. Stochastic filtering in a probabilistic action model. In *AAAI'07*, 2007.

[9] Michael Jordan. *Introduction to probabilistic graphical models*. 2007. Forthcoming.

[10] K. Kersting, L. D. Raedt, and T. Raiko. Logical hidden markov models. *Artificial Intelligence Research*, 2006.

[11] U. Kjaerulff. A computational scheme for reasoning in dynamic probabilistic networks. In *UAI*, 1992.

[12] S. Muggleton. Stochastic logic programs. In *Workshop on ILP*, 1995.

[13] Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, 2002.

[14] M. Nance, A. Vogel, and E. Amir. Reasoning about partially observed actions. In *AAAI*, 2006.

[15] D. Pool. Probabilistic horn abduction and bayesian networks. *Artificial Intelligence*, 64:81–129, 1993.

[16] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *IEEE*, 1989.

[17] R. Reiter. *Knowledge In Action*. MIT Press, 2001.

[18] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 2006.

[19] A. Shirazi and E. Amir. First order logical filtering. In *IJCAI*, 2005.

[20] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *UAI'02*, 2002.

[21] M. Welling, I. Porteous, and E. Bart. Infinite state bayesian networks. In *NIPS*, 2007.

[22] L. S. Zettlemoyer, H. M. Pasula, and L. P. Kaelbling. Logical particle filtering. In *Dagstuhl Seminar on Probabilistic, Logical, and Relational Learning*, 2007.

Sparse Stochastic Finite-State Controllers for POMDPs

Eric A. Hansen
Dept. of Computer Science and Engineering
Mississippi State University
Mississippi State, MS 39762
hansen@cse.msstate.edu

Abstract

Bounded policy iteration is an approach to solving infinite-horizon POMDPs that represents policies as stochastic finite-state controllers and iteratively improves a controller by adjusting the parameters of each node using linear programming. In the original algorithm, the size of the linear programs, and thus the complexity of policy improvement, depends on the number of parameters of each node, which grows with the size of the controller. But in practice, the number of parameters of a node with non-zero values is often very small, and does not grow with the size of the controller. Based on this observation, we develop a version of bounded policy iteration that leverages the sparse structure of a stochastic finite-state controller. In each iteration, it improves a policy by the same amount as the original algorithm, but with much better scalability.

1 Introduction

Partially observable Markov decision processes (POMDPs) provide a framework for decision-theoretic planning problems where actions need to be taken based on imperfect state information. Many researchers have shown that a policy for an infinite-horizon POMDP can be represented by a finite-state controller. In some cases, this is a deterministic controller in which a single action is associated with each node, and an observation results in a deterministic transition to a successor node (Kaelbling, Littman, & Cassandra, 1998; Hansen, 1998; Meuleau, Kim, Kaelbling, & Cassandra, 1999a). In other cases, it is a stochastic controller in which actions are selected based on a probability distribution associated with each node, and an observation results in a probabilistic transition to a successor node (Platzman, 1981; Meuleau, Peshkin, Kim, & Kaelbling, 1999b; Baxter & Bartlett, 2001; Poupart & Boutilier, 2004; Amato, Bernstein, & Zilberstein, 2007).

Bounded policy iteration (BPI) is an approach to solving infinite-horizon POMDPs that represents policies as stochastic finite-state controllers and iteratively improves a controller by adjusting the parameters of each node using linear programming, where the parameters specify the action selection and node transition probabilities of the node (Poupart & Boutilier, 2004). BPI is related to an exact policy iteration algorithm for POMDPs due to Hansen (1998), but differs from it by providing an elegant and effective approach to approximation in which bounding the size of the controller allows a tradeoff between planning time and plan quality. Originally developed as an approach to solving single-agent POMDPs, BPI has also been generalized for use in solving decentralized POMDPs (Bernstein, Hansen, & Zilberstein, 2005).

In BPI, the complexity of policy improvement depends on the size of the linear programs used to adjust the parameters of each node of the controller. In turn, this depends on the number of parameters of each node (as well as the size of the state space). In the original algorithm, each node of the controller has $|A| + |A||Z||\mathcal{N}|$ parameters, where $|A|$ in the number of actions, $|Z|$ is the number of observations, and $|\mathcal{N}|$ is the number of nodes of the controller. This assumes a fully-connected stochastic controller. In practice, however, most of these parameters have zero probabilities, and the number of parameters of a node with non-zero probabilities remains relatively constant as the number of nodes of the controller increases. Based on this observation, we propose a modified version of BPI that leverages a sparse representation of a stochastic finite-state controller. In each iteration, it improves the controller by the same amount as the original algorithm. But it does so by solving much smaller linear programs, where the number of variables in each linear program depends on the number of parameters of a node with non-zero values. Because the number of parameters of a node with non-zero values tends to remain relatively constant as the size of the controller grows, the complexity of each iteration of the modified version of BPI tends to grow only linearly with the number of nodes of a controller, which is a dramatic improvement in scalability compared to the original algorithm.

2 Background

We consider a discrete-time infinite-horizon POMDP with a finite set of states, $\mathcal{S}$, a finite set of actions, $\mathcal{A}$, and a finite set of observations, $\mathcal{Z}$. Each time period, the environment is in some state $s \in \mathcal{S}$, the agent takes an action $a \in \mathcal{A}$, the environment makes a transition to state $s' \in \mathcal{S}$ with probability $P(s'|s,a)$, and the agent observes $z \in Z$ with probability $P(z|s',a)$. In addition, the agent receives an immediate reward with expected value $R(s,a) \in \Re$. We assume the objective is to maximize expected total discounted reward, where $\beta \in (0,1]$ is the discount factor.

Since the state of the environment cannot be directly observed, we let b denote an $|\mathcal{S}|$-dimensional vector of state probabilities, called a *belief state*, where $b(s)$ denotes the probability that the system is in state s. If action a is taken and followed by observation z, the successor belief state, denoted b^a_z, is determined using Bayes' rule.

2.1 Policy representation and evaluation

A *policy* for a POMDP can be represented by a finite-state controller (FSC). A stochastic finite-state controller is a tuple $< \mathcal{N}, \psi, \eta >$ where $\mathcal{N}$ is a finite set of nodes, ψ is an action selection function that specifies the probability, $\psi_n(a) = P(a|n)$, of selecting action $a \in A$ when the FSC is in node $n \in \mathcal{N}$, and η is a node transition function that specifies the probability, $\eta_n(a,z,n') = P(n'|n,a,z)$, that the FSC will make a transition from node $n \in \mathcal{N}$ to node $n' \in \mathcal{N}$ after taking action $a \in A$ and receiving $z \in Z$.

The value function of a policy represented by a FSC is piecewise linear and convex, and can be computed exactly by solving the following system of linear equations, with one equation for each pair of node $n \in \mathcal{N}$ and state $s \in S$:

$$V_n(s) = \sum_{a \in A} \psi_n(a) R(s,a) + \beta \sum_{a,z,s,n'} \eta_n(a,z,n') P(s'|s,a) P(z|s',a) V_{n'}(s'). \quad (1)$$

In this representation of the value function, there is one $|S|$-dimensional vector V_n for each node $n \in \mathcal{N}$ of the controller. The value of any belief state b is determined as follows,

$$V(b) = \max_{n \in N} \sum_{s \in S} b(s) V_n(s), \quad (2)$$

and the controller is assumed to start in the node that maximizes the value of the initial belief state. The value function of an optimal policy satisfies the optimality equation,

$$V(b) = \max_{a \in A} \left\{ R(b,a) + \beta \sum_{z \in Z} P(z|b,a) V(b^a_z) \right\}, \quad (3)$$

where $R(b,a) = \sum_{s \in S} b(s) R(s,a)$ and $P(z|b,a) = \sum_{s \in S} b(s) \sum_{s' \in S} P(s'|s,a) P(z|s',a)$.

Variables: ϵ; $\psi_n(a), \forall a$; $\eta_n(a,z,n'), \forall a,z,n'$
Objective: Maximize ϵ
Improvement constraints:
$V_n(s) + \epsilon \leq \sum_a \psi_n(a) R(s,a) +$
$\gamma \sum_{a,z,n'} \eta_n(a,z,n') P(s'|s,a) P(z|s',a) V_{n'}(s'), \forall s$
Probability constraints:
$\sum_a \psi(a) = 1$
$\sum_{n'} \eta_n(a,z,n') = \psi(a), \forall a,z$
$\psi(a) \geq 0, \forall a$
$\eta_n(a,z,n') \geq 0, \forall a,z,n'$

Table 1: Linear program for improving a node n of a stochastic finite-state controller.

2.2 Bounded policy iteration

Policy iteration algorithms iteratively improve a policy by alternating between two steps: policy evaluation and policy improvement. Hansen (1998) proposed a policy iteration algorithm for POMDPs that represents a policy as a deterministic finite-state controller. In the policy improvement step, it uses the dynamic programming update for POMDPs to add, merge and prune nodes of the controller. The algorithm is guaranteed to converge to an ϵ-optimal controller and can detect convergence to an optimal policy.

A potential problem is that the number of nodes added in the policy improvement step can be large, and the controller can grow substantially in size from one iteration to the next. Because the complexity of the policy improvement step increases with the size of the controller, allowing the size of the controller to grow too fast can slow the rate of further improvement and limit scalability. The same problem occurs in value iteration, where the number of vectors in the piecewise linear and convex representation of the value function can grow too fast from one iteration to the next. Feng and Hansen (2001) describe a method for performing approximate dynamic programming updates that has the effect of reducing the number of vectors added to a value function in value iteration, or the number of nodes added to a controller in policy iteration.

For policy iteration, Poupart and Boutilier (2004) propose another approach that also has the benefit of avoiding an expensive dynamic programming update. Their *bounded policy iteration* algorithm controls growth in the size of the controller in two ways. First, a policy is represented as a stochastic finite-state controller that can be improved without increasing its size, by adjusting the action and node-transition probabilities of each node of the controller using linear programming. Second, when the controller cannot be improved further in this way, k nodes are added to the controller, where k is some small number greater than or equal to one. In the rest of this background section, we review each of these two steps in further detail.

Algorithm 1 Bounded policy iteration

repeat
  repeat
    {Policy evaluation}
    Solve the linear system given by Equation (1)
    {Policy improvement without adding nodes}
    for each node n of the controller **do**
      solve the linear program in Table 1
      if $\epsilon > 0$ **then**
        update parameters and value vector of node
      end if
    end for
  until no improvement of controller
  {Policy improvement by adding nodes}
  find belief states reachable from tangent points
  create k new nodes that improve their value
until no new node is created

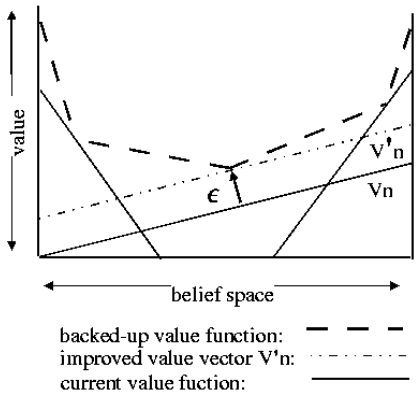


Figure 1: BPI can improve the value vector V_n by an amount ϵ to obtain the improved value vector V_n', which is tangent to the backed-up value function.

Improving nodes The first step attempts to improve a controller while keeping its size fixed. For each node n of the controller, the linear program in Table 1 is solved. The linear program searches for action probabilities and node transition probabilities for the node that improve the value vector V_n associated with the node by some amount ϵ for each state, where ϵ is the objective maximized by the linear program. If an improvement is found, the parameters of the node are updated accordingly. (The value vector may also be updated, and will be further improved during the policy evaluation step.)

Poupart and Boutilier (2004) show that the linear program can be interpreted as follows: it implicitly considers the vectors of the backed-up value function that would be created by performing a dynamic programming update. In particular, the linear program searches for a convex combination of these backed-up vectors that pointwise dominates the value vector currently associated with the node. If an improvement is found, the parameters of the convex combination become the new parameters of the node. This interpretation is illustrated in Figure 1, which is adapted from a similar figure from Poupart's dissertation (2005). As Figure 1 shows, the new value vector is parallel to the old vector (i.e., the value of each component is improved by same amount) and it is tangent to the backed-up value function.

Adding nodes Eventually, no node can be improved further by solving its corresponding linear program. At this point, BPI is at a local optimum. It can escape such a local optimum by adding one or more nodes to the controller.

The key insight is that when a local optimum is reached, the value vector of each node of the controller is tangent to the backed-up value function at one or more belief states. Moreover, the solution of the linear program that is the dual of the linear program in Table 1 is a belief state that is tangent to the backed-up value function, and so is called the *tangent belief state*. (Since most linear program solvers return both the primal and dual solutions when they solve a linear program, we assume that we get the tangent belief state when we solve the linear program in Table 1, in addition to the value of ϵ and the action and node transition probabilities.)

To escape a local optimum, it is necessary to improve the value of the tangent belief state. This leads to a method for adding nodes to a controller. Given a tangent belief state b, the algorithm considers every belief state b' that can be reached from it in one step (i.e., by taking some action a followed by some observation z). For each reachable belief state, a backup is performed (as defined by Equation 3). If the backed-up value is better than the value of the belief state based on the current value function, a deterministic node is added to the controller that has the same action, and, for each observation, the same successor node, that created the improved backed-up value. Often, it is only necessary to add a single node to the controller to escape a local optimum. But because a value vector may be tangent to the backed-up value function for a linear portion of belief space, it may be necessary to add more than one node to escape the local optimum. As we will see, this method for adding nodes to escape local optima is related to the approach that we develop in this paper.

Two sources of complexity Most of the computation time of BPI is spent in solving the linear programs that are used to adjust the parameters that specify the action selection probabilities and node transition probabilities of each node of a controller. The size of the linear programs, and thus the complexity of BPI, depends on two things: the size of the controller (which determines the number of variables in the linear program) and the size of the state space (which determines the number of constraints).

Test problem	Statistic	Number of nodes of controller 50	100	150	200	250	300
Slotted Aloha $\|S\| = 30, \|A\| = 9, \|Z\| = 3$	total parameters per node	1,359	2,709	4,059	5,409	6,759	8,109
	min non-zero parameters	6	5	7	7	7	6
	avg non-zero parameters	11	11	11	11	11	11
	max non-zero parameters	17	18	20	21	21	21
Tiger grid $\|S\| = 36, \|A\| = 5, \|Z\| = 17$	total parameters per node	4,255	8,505	12,755	17,010	21,255	25,505
	min non-zero parameters	45	34	37	33	31	32
	avg non-zero parameters	68	67	68	68	69	68
	max non-zero parameters	98	97	97	114	98	114
Hallway $\|S\| = 60, \|A\| = 5, \|Z\| = 21$	total parameters per node	5,255	10,505	15,755	21,005	26,255	31,505
	min non-zero parameters	28	44	32	35	36	35
	avg non-zero parameters	99	112	105	103	102	100
	max non-zero parameters	130	151	158	157	158	158
Hallway2 $\|S\| = 92, \|A\| = 5, \|Z\| = 17$	total parameters per node	4,255	8,505	12,755	17,010	21,255	25,505
	min non-zero parameters	46	35	29	39	37	22
	avg non-zero parameters	91	105	112	109	109	114
	max non-zero parameters	144	155	163	166	164	167

Table 2: Total number of parameters per node of stochastic finite-state controllers found by bounded policy iteration, and minimum, average, and maximum number of parameters with non-zero values, as a function of the size of the controller. (The average is rounded up to the nearest integer.) The four test POMDPs are from (Cassandra, 2004).

In this paper, we focus on the first source of complexity, that is, we focus on improving the complexity of BPI with respect to controller size. (By controller size, we mean not only the number of nodes of the controller, but also the number of actions and observations, since these together determine the number of parameters of a node.) Coping with POMDPs with large state spaces is an orthogonal research issue, and several approaches have been developed that can be combined with BPI. For example, Poupart and Boutilier (2005) describe a technique called *value-directed compression* and report that it allows BPI to solve POMDPs with millions of states. Because the test problems used in this paper have small state spaces, it is important to keep in mind that the techniques developed in the next section for improving the scalability of BPI with respect to controller size can be combined with techniques for coping with large state spaces.

3 Sparse bounded policy iteration

In this section, we describe a modified version of BPI that we call *Sparse BPI*. In each iteration, it improves a FSC by the same amount as the original algorithm, but with much improved scalability. To motivate our approach, we begin with a discussion of the sparse structure of stochastic finite-state controllers found by BPI.

3.1 Sparse stochastic finite-state controllers

As we have seen, each iteration of BPI solves $|\mathcal{N}|$ linear programs, and each linear program has $|A| + |A||Z||\mathcal{N}|$ variables and $|S| + |A||Z|$ constraints (in addition to the constraints that the variables have non-negative values). Even for small FSCs, the number of variables in the linear programs can be very large, and the fact that the number of variables grows with the number of nodes in the controller significantly limits the scalability of BPI.

If we look at the controllers produced by BPI, however, we find that most of the parameters of each node (i.e., most of the variables in the solutions of the linear program) have values of zero. Table 2 illustrates this vividly for four benchmark POMDPs from (Cassandra, 2004). The overwhelming majority of each node's parameters have zero probabilities. This is despite the fact that *all* of the nodes of these controllers are stochastic. Note that a deterministic node has $1 + |Z|$ non-zero probabilities, one for a choice of action, and $|Z|$ to specify the successor node for each observation. Table 2 shows that the minimum number of non-zero parameters for any node is always greater than this, which indicates that all of the nodes are stochastic. For these problems and many others, BPI is very effective in leveraging the possibility of stochastic nodes to improve a controller without increasing its size. Nevertheless, the actual number of parameters with non-zero probabilities is a very small fraction of the total number of parameters.

Besides the sparsity of the stochastic finite-state controllers found by BPI, an equally important observation about the results in Table 2 is that the number of parameters with non-zero probabilities tends to remain the same as the controller grows in size, whereas the total number of parameters grows significantly. In the following, we develop a modified version of BPI that exploits this sparse structure.

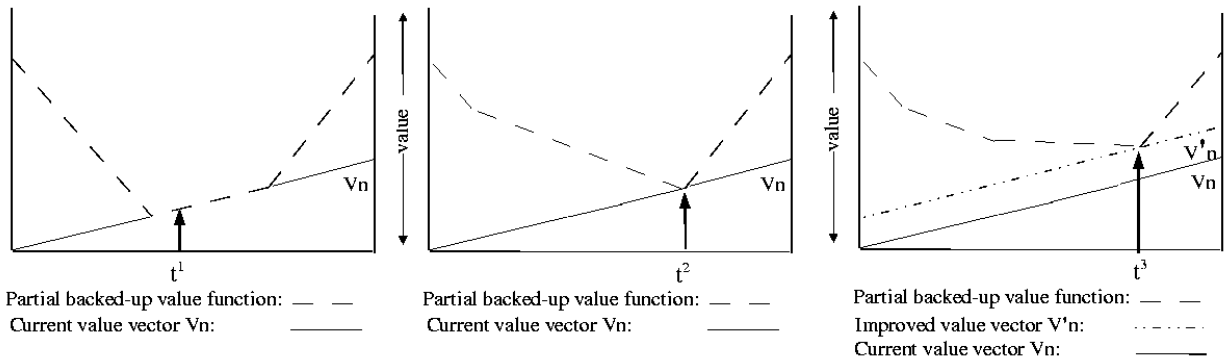


Figure 2: In the left panel, the value vector is tangent to the partial backed-up value function along a linear segment. The center panel shows the result of adding parameters to the sparse linear program that improve the partial backed-up value function at tangent belief state t^1. Although this does not improve the value vector, it creates a new tangent belief state t^2. When parameters are added to the sparse linear program that improve its backed-up value, the result is an improved value vector with a new tangent belief state t^3, as shown in the right panel.

3.2 Sparse policy improvement algorithm

We begin by describing the main idea of the algorithm. We have seen that the number of parameters of a node with non-zero probabilities in the solution of the linear program in Table 1 is very small relative to the total number of parameters. Let us call these the *useful parameters*. If we could somehow identify the useful parameters of a node, we could improve a node by solving a linear program that only includes these variables. We call a linear program that only includes some of the parameters of a node a *reduced linear program*. Our approach will be to solve a series of reduced linear programs, where the last one is guaranteed to include all of the useful parameters of the node. This approach will achieve the same result as solving the full linear program of Table 1, but will be more efficient if the reduced linear programs are much smaller.

We next describe an iterative method for identifying a small set of parameters that includes all of the useful parameters. Our method starts with the set of parameters of the node that currently have non-zero probabilities. This guarantees a solution that is at least as good as the current policy for the node. Then we add parameters using a technique that is inspired by the technique BPI uses to add nodes to a controller in order to escape local optima. As shown in Figure 1, the value vector that is created by solving the linear program in Table 1 is tangent to the backed-up value function, and the solution of the dual linear program is a tangent belief state. In the case of a reduced linear program, however, we have a *partial backed-up value function* that does not include vectors corresponding to parameters that are not included in the reduced linear program. (For example, if only one action parameter is included in the reduced linear program, the partial backed-up value function does not include any vectors created by taking a different action.)

If the reduced linear program is missing some useful parameters, it must be possible to improve the value of the tangent belief state b by considering the vectors of the full backed-up value function. So, to identify potentially useful parameters, we perform a backup for the tangent belief state b. If the backed-up value, which is defined as

$$\max_{a \in A} \left\{ R(b,a) + \beta \sum_{z \in Z} P(z|b,a) \max_{n \in N} \left(\sum_{s \in S} b_z^a(s) V_n(s) \right) \right\},$$

is greater than the value of the tangent belief state based on the current value vector $V_{n'}$, which is defined as $\sum_{s \in S} b(s) V_{n'}(s)$, we add to our set of parameters the parameters used to create the backed-up value: the best action a, and, for each observation z, the best successor node n. Then we solve another linear program that includes these parameters in addition to the parameters contained in the previous reduced linear program.

If the value vector is tangent to the partial backed-up value function at a single belief state, adding the parameters that improve the backed-up value of this tangent belief state improves the value vector for this node. But since the value vector may be tangent to the partial backed-up value function along a linear segment, adding parameters does not guarantee improvement of the node. In this case, however, it does change the result of the new reduced linear program in an important way. Because adding these parameters to the reduced linear program improves the partial backed-up value function that is searched by the linear program, the value vector is no longer tangent to the partial backed-up value function at the same belief state. In other words, even if the solution of the reduced linear program is not changed by adding these parameters, the solution of its dual linear program changes in one important way: there must be a new tangent belief state. This is illustrated in Figure 2.

This points to an iterative algorithm for improving a node. At each step, the algorithm solves a reduced linear program, and then performs a backup for the tangent belief state. If the backed-up value of the tangent belief state is better than its value based on the current value vector, the parameters that produced the improved backed-up value are added to the reduced linear program. Since the backed-up value of the tangent belief state is improved, it must be the case that at least some of these parameters are not in the current linear program; if they were, the current value vector would not be tangent to the partial backed-up value function at this belief state. The condition for terminating this iterative process of adding parameters to a reduced linear program is based on the following lemma.

Lemma 1 *The difference between the backed-up value of a tangent belief state and its value based on the current value vector bounds the amount by which the linear program in Table 1 can improve the value vector of a node.*

Proof: The linear program in Table 1 implicitly uses the backed-up value function to improve the value vector associated with a node by an amount ϵ for each state value of the vector; in turn, this improves by ϵ the value of any belief state based on the value vector. So, if the difference between the backed-up value of any belief state and its value based on the current value vector is δ, then $\epsilon \leq \delta$.□

It follows that if the backed-up value of any belief state is not an improvement of its value based on the current value vector, the linear program cannot improve the value vector.

Based on this lemma, we can prove the following.

Theorem 1 *This procedure of improving a stochastic node by solving a sequence of reduced linear programs is guaranteed to terminate and the last reduced linear program in the sequence produces the same result as solving the full linear program in Table 1.*

Proof: The procedure is guaranteed to terminate because whenever the backed-up value of the tangent belief state is greater than its value based on the value vector created by solving the current reduced linear program, at least one (and no more than $(1 + |Z|)$ parameters will be added to the linear program, and the total number of parameters is finite.

To see that the last reduced linear program solved in this sequence has the same result as solving the full linear program in Table 1, note that the procedure terminates when the difference between the backed-up value of the tangent belief state (for the last reduced linear program) and the value of the tangent belief state based on the best value vector found so far for this node is zero. From Lemma 1, it follows that no further improvement is possible.□

We call this iterative approach to improving the nodes of a stochastic FSC *sparse policy improvement*. The high-level pseudocode is shown in Algorithm 2. Although it can solve multiple linear programs for each linear program solved by the original BPI algorithm, it has an advantage if the number of variables in each of these reduced linear programs is very small compared to the number of variables in the full linear program of Table 1. This is the case whenever the FSCs found by BPI are very sparse.

Algorithm 2 Sparse policy improvement

```
for each node of controller do
  Create initial reduced linear program
  {its variables correspond to the parameters of the node
  that currently have non-zero probabilities}
  threshold ← 0
  repeat
    Solve the reduced linear program
    if ε > threshold then
      Use solution of LP to update parameters of node
      threshold ← ε
    end if
    Do backup for tangent belief state
    if backed-up value > (current value + ε) then
      Add variables to linear program that correspond
      to parameters that produced backed-up value
    end if
  until no new variables are added to linear program
  increase value vector of node by ε
end for
```

As mentioned before, this algorithm is inspired by the way the original BPI algorithm adds nodes to a controller. In both cases, the technique for breaking a local optimum at a tangent belief state is to improve the backed-up value function of the tangent belief state. In the original BPI algorithm, the value of the tangent belief state is improved by adding nodes to the controller. In sparse policy improvement, it is improved by adding parameters to a node.

There is also an interesting relationship between this algorithm and column generation methods in linear programming (Bertsekas & Tsitsiklis, 1997). Column generation is a useful strategy for solving linear programs where the number of variables is very large, the number of constraints is relatively small, and most variables have a value of zero in the optimal solution. It begins by considering a small subset of the variables (i.e., the columns) of a problem, and solves a reduced linear program with only these variables. Then it identifies unused variables that can be added to the linear program to improve the result. Use of this strategy requires some way of determining if the current solution is optimal, and if it is not, some way of generating one or more unused variables that can improve the solution. Our algorithm can be viewed as an application of this general strategy to the problem of solving POMDPs using BPI.

Test problem	Algorithm	Number of nodes of controller 50	100	150	200	250	300
Slotted Aloha	BPI	202	415	684	871	1,085	1,620
$\|S\| = 30, \|A\| = 9, \|Z\| = 3$	Sparse-BPI	16	19	19	20	21	23
Tiger grid	BPI	1,365	3,447	5,822	9,461	12,358	15,770
$\|S\| = 36, \|A\| = 5, \|Z\| = 17$	Sparse-BPI	189	212	335	174	207	274
Hallway	BPI	3,900	10,180	17,340	24,485	27,428	32,973
$\|S\| = 60, \|A\| = 5, \|Z\| = 21$	Sparse-BPI	1,215	935	1,110	973	1,094	1,267
Hallway2	BPI	7,760	17,729	29,809	42,905	53,903	68,898
$\|S\| = 92, \|A\| = 5, \|Z\| = 17$	Sparse-BPI	2,668	4,128	3,323	4,078	3,513	3,388

Table 3: Average time (in CPU milliseconds) for improving a single node of a finite-state controller, as a function of the size of the controller, for four benchmark POMDPs.

4 Experimental results

We implemented sparse bounded policy iteration and tested it on several benchmark POMDPs. Table 3 shows results for the four POMDPs considered earlier in Table 2. The experiments ran on a 3.0 GHz processor using a linear program solver in CPLEX version 9.0. We solved each POMDP beginning with an initial controller with $|A|$ nodes, and adding up to five new nodes each time the policy improvement step reached a local optimum. Table 3 reports the average running time to improve a single node because this dominates overall computation time, and also because it measures performance independently of design decisions such as how and when to add nodes to a controller.

Table 3 shows that Sparse BPI is much more efficient than the original BPI algorithm in improving sparse stochastic FSCs. Even for relatively small controllers of 300 nodes, Sparse BPI runs between 20 and 80 times faster than BPI in solving these particular POMDPs. More importantly, its relative advantage grows with the size of the controller. The size of the reduced linear programs solved by Sparse BPI remains about the same as the controller grows in size; as a result, the time it takes to improve a single node of the controller remains relatively constant as the size of the controller grows, in contrast to BPI.

An interesting difference between Sparse BPI and BPI is that the running time of an iteration of Sparse BPI depends on how much improvement of the controller is possible. If the nodes of a controller are already at a local optimum, Sparse BPI often needs to solve only a couple reduced linear programs per node in order to determine that further improvement is not possible. In this case, an iteration of Sparse BPI terminates relatively quickly. But if much improvement of the FSC is possible, Sparse BPI often needs to solve ten or twenty or even more reduced linear programs for some nodes in order to add all of the parameters that are needed to maximize improvement of the node. To obtain reliable running times for Sparse-BPI, the results in Table 3 are averaged over several iterations of Sparse-BPI.

Table 3 shows that a larger state space slows both algorithms, but it slows the sparse algorithm more. This is because the sparse algorithm solves multiple linear programs for each linear program solved by the original algorithm, and more constraints makes these more difficult to solve.

The running time of Sparse BPI could be reduced further by finding a way to reduce the number of iterations it takes to improve a node. For the *hallway2* problem, for example, the sparse algorithm can take ten or more iterations to improve a node – occasionally, as many as thirty or forty. Each iteration requires solving a reduced linear program. But in fact, it is only necessary to solve one linear program to change the parameters of a node. The other linear programs in the sequence of reduced linear programs solved by the sparse algorithm are used to identify a sequence of tangent belief states; for each tangent belief state, a backup is performed in order to identify parameters to add to the stochastic node. It seems possible to identify a set of belief states for which to perform backups without needing to solve a linear program to identify each one.

Point-based methods solve POMDPs by performing a backup for each of a finite set of belief states (Pineau, Gordon, & Thrun, 2003). Recent work shows that this approach can be combined with policy iteration algorithms that represent a policy as a finite-state controller (Ji, Parr, Li, Liao, & Carin, 2007). An interesting direction of research is to use point-based backups to identify parameters that can be added to nodes; after many backups are performed, a reduced linear program could be solved for each node in order to improve its action selection and node transition probabilities – integrating point-based methods with policy iteration for stochastic finite-state controllers.

Another possible way to reduce the number of iterations is early termination. Instead of continuing to add parameters until the difference between the backed-up value of the tangent belief state and its value based on the current value vector is zero, Sparse-BPI could stop adding parameters as soon as the difference is small enough to demonstrate that only a small amount of further improvement is possible.

5 Conclusion

We have presented a modified bounded policy iteration algorithm for POMDPs called sparse bounded policy iteration. The new algorithm exploits the sparse structure of stochastic finite-state controllers found by bounded policy iteration. Each iteration of the algorithm produces the identical improvement of a controller that an iteration of the original bounded policy iteration algorithm produces, but with improved scalability. Whereas the time it takes for the original algorithm to improve a single node grows with the size of the controller, the time it takes for the new algorithm to improve a single node is typically independent of the size of the controller. This makes it practical to use bounded policy iteration to find larger controllers for POMDPs.

Acknowledgements

This work was supported in part by the Air Force Office of Scientific Research under Award No. 05-003202A05.

References

Amato, C., Bernstein, D., & Zilberstein, S. (2007). Solving POMDPs using quadratically constrained linear programs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pp. 2418–2424.

Baxter, J., & Bartlett, P. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, *15*, 319–350.

Bernstein, D., Hansen, E., & Zilberstein, S. (2005). Bounded policy iteration for decentralized POMDPs. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence(IJCAI-05)*, pp. 1287–1292.

Bertsekas, D., & Tsitsiklis, J. (1997). *Introduction to Linear Optimization*. Athena Scientific.

Cassandra, A. (2004). Tony's POMDP file repository page.. http://pomdp.org/pomdp/examples/index.shtml.

Feng, Z., & Hansen, E. (2001). Approximate planning for factored POMDPs. In *Proceedings of the 6th European Conference on Planning*.

Hansen, E. (1998). Solving POMDPs by searching in policy space. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pp. 211–219.

Ji, S., Parr, R., Li, H., Liao, X., & Carin, L. (2007). Point-based policy iteration. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI-07)*, pp. 1243–1249.

Kaelbling, L., Littman, M., & Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*, 99–134.

Meuleau, N., Kim, K., Kaelbling, L., & Cassandra, A. (1999a). Solving POMDPs by searching the space of finite policies. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 417–426.

Meuleau, N., Peshkin, L., Kim, K., & Kaelbling, L. (1999b). Learning finite-state controllers for partially observable environments. In *Proceedings ofthe 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 427–436.

Pineau, J., Gordon, G., & Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pp. 1025–1032.

Platzman, L. (1981). A feasible computational approach to infinite-horizon partially-observed Markov decision problems. Tech. rep., Georgia Institute of Technology. Reprinted in Working Notes of the AAAI Fall Symposium on Planning using Partially Observable Markov Decision Processes, 1998.

Poupart, P. (2005). *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. Ph.D. thesis, University of Toronto.

Poupart, P., & Boutilier, C. (2004). Bounded finite state controllers. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, pp. 823–830. MIT Press.

Poupart, P., & Boutilier, C. (2005). VDCBPI: An approximate scalable algorithm for large POMDPs. In *Advances in Neural Information Processing Systems 17: Proceedings of the 2004 Conference*. MIT Press.

Convergent Message-Passing Algorithms for Inference over General Graphs with Convex Free Energies

Tamir Hazan **Amnon Shashua**

School of Eng. & Computer Science
The Hebrew University of Jerusalem
Israel

Abstract

Inference problems in graphical models can be represented as a constrained optimization of a free energy function. It is known that when the Bethe free energy is used, the fixed-points of the belief propagation (BP) algorithm correspond to the local minima of the free energy. However BP fails to converge in many cases of interest. Moreover, the Bethe free energy is non-convex for graphical models with cycles thus introducing great difficulty in deriving efficient algorithms for finding local minima of the free energy for general graphs. In this paper we introduce two efficient BP-like algorithms, one sequential and the other parallel, that are guaranteed to converge to the global minimum, for any graph, over the class of energies known as "convex free energies". In addition, we propose an efficient heuristic for setting the parameters of the convex free energy based on the structure of the graph.

1 Introduction

Probabilistic graphical models present a convenient and popular tool for reasoning about complex distributions. The graphical model reflects the way the complex distribution factors into a product of distributions over a small number of variables (cliques), where the graph represents the incidence between cliques and the variables contained in them — such a graph is known as a *factor graph*. The probabilistic inference is represented by a way of calculating marginal distributions (or the most likely assignment of variables) efficiently using the structure of the graph.

One of the most popular class of methods for inference over (factor) graphs are message-passing algorithms which pass messages along the edges of the factor graph until convergence. The belief propagation (BP) algorithm (and its extensions) is popular, and has received the most attention, due to its simplicity and computational efficiency. The BP algorithm is exact, i.e., the resulting marginal distributions are the correct ones, when the factor graph is free of cycles. An intriguing feature of BP, which most likely is the source for its great popularity, is that it often gives surprisingly good approximate results for graphical models with cycles. However, in this context there are no convergence guarantees (except under some special cases (Mooij & Kappen, 2005)) and the algorithm often fails to converge.

It is known that the fixed-points of the BP algorithm correspond to local minima of a constrained energy function called the Bethe free energy (Yedidia et al., 2005). The free energy arises from the expansion of the KL-divergence between the input distribution and its product form. The Bethe function replaces the entropy term in the free energy by an approximation which is exact for factor graphs without cycles. In such a case, the Bethe free energy is convex over the set of constraints (representing validity of marginals). When the factor graph has cycles the Bethe energy is non-convex and although it is possible to derive convergent algorithms to a local minima of the Bethe function (Yuille, 2002) the computational cost is large and thus has not gained popularity.

To overcome the difficulty with the non-convexity of the Bethe approximation, several authors have introduced a class of approximations known as *convex free energies* which are convex over the set of constraints for any factor graph. An important member of this class is the tree-reweighted (TRW) free energy which consist of a linear combination of free energies defined on spanning trees of the factor graph. It is notable that for this specific member of convex free energies a convergent message-passing algorithm has been recently introduced (Globerson & Jaakkola, 2007b). The algorithm is sequential (unlike BP which has both se-

quential and parallel forms) and applies to graphs with pairwise cliques only. However, a convergent message passing algorithm for the general class of convex free energies is still lacking. The existing algorithms either employ damping heuristics to ensure convergence in practice (Wainwright et al., 2005) or focus on a sub-class of free energies where the entropy term is a positive combination of joint entropies (Heskes, 2006).

In this paper, we derive convergent message-passing algorithms, one sequential and the other parallel, for the general class of convex free energies. The derivation applies to general factor graphs (cliques of any size) and have a similar architecture to the BP algorithm. The algorithms are based on a general framework for handling optimization problems of the type $f(b) + \sum_i h_i(b)$ where $f(b)$ is strictly convex and $h_i(b)$ are convex but *not necessarily strict nor differentiable*. We show that problems of this class have a simple block-update (sequential and parallel) message-passing solution. We then map the constrained convex free energy problem into this framework.

Independently, we propose also a heuristic for setting up the parameters of the convex free energy from the structure of the factor graph. The key idea is to strive for a convex free energy which is as close as possible to Bethe's free energy under a set of constraints governing the class of convex free energies. The underlying motivation is borne by the empirical observation from BP practitioners that when BP does converge, the results are often surprisingly good (Murphy et al., 1999). Since our scheme would always converge and the free energy approximation is close to Bethe's, we would have in some sense a "convergent BP" for general graphs.

2 Terminology and Problem Setup

We consider a joint distribution $P(\mathbf{x})$ on a set of discrete variables $\mathbf{x} = x_1, ..., x_n$ in a finite domain. In a graphical model we suppose that $P(\mathbf{x})$ factors into a product of non-negative functions (clique potentials):

$$P(\mathbf{x}) = \frac{1}{Z} \prod_\alpha \psi_\alpha(\mathbf{x}_\alpha),$$

where α is an index labeling m functions $\psi_1, ..., \psi_m$ and the function $\psi_\alpha(\mathbf{x}_\alpha)$ has arguments $\mathbf{x}_\alpha$ that are some subset of $\{x_1, ..., x_n\}$ and Z is a normalization constant. The factorization structure above is conveniently represented by a *factor graph* (Kschischang et al., 2001) which is a bipartite graph with variable nodes one for each variable x_i and a factor node for each function ψ_α. An edge connects a variable node i with factor node α if and only if $x_i \in \mathbf{x}_\alpha$, i.e., x_i is an argument of ψ_α. We adopt the terminology where $N(i)$ stands for all factor nodes that are neighbors of variable node i and $N(\alpha)$ stands for all variable nodes that are neighbors of factor node α. Finally, we limit our treatment to factor graphs where any pair of factors intersect in at most a single variable node, i.e., $N(\alpha) \cap N(\beta)$ is either empty or equal to some i. There is no technical limitation to allow for higher-order intersections but that comes at the price of reducing the clarity of our presentation. In Section 7 we will explain the necessary additions for handling general factor graphs.

The typical task we try to perform is to compute the marginal distributions $P(x_i) = \sum_{\mathbf{X} \setminus x_i} P(\mathbf{x})$ and $P(\mathbf{x}_\alpha) = \sum_{\mathbf{X} \setminus x_\alpha} P(\mathbf{x})$. Basically, the computation requires the summation over the states of all the variable nodes not in $\mathbf{x}_\alpha$ (including the case of the singleton x_i). This computation is generally hard because it can require summing up exponentially large number of terms — thus one seeks efficient ways or approximate solutions for the marginals.

Let $b(\mathbf{x})$ stand for the approximate distribution where $b_i(x_i)$ approximates $P(x_i)$ and $b_\alpha(\mathbf{x}_\alpha)$ approximates $P(\mathbf{x}_\alpha)$ with the constraints that $\sum_{\mathbf{x}_\alpha \setminus x_i} b_\alpha(\mathbf{x}_\alpha) = b_i(x_i)$ for all $\alpha \in N(i)$. The *free energy* arises from minimizing the KL-divergence between the approximate distribution $b(\mathbf{x})$ and the un-normalized product form:

$$D(b \mid\mid \prod_\alpha \psi_\alpha(\mathbf{x}_\alpha)) = \sum_\alpha \sum_{\mathbf{x}_\alpha} E_\alpha(\mathbf{x}_\alpha) b_\alpha(\mathbf{x}_\alpha) - H(b),$$

where $E_\alpha = -\ln \psi_\alpha$ and $H(b)$ is the entropy of $b(\mathbf{x})$. In other words, the free energy consists of a sum of a linear term over b which is exponential in the size of the cliques and and an entropy term (which is exponential in n). The approximate methods for computing the marginals are based on choosing an approximation to the entropy term $H(b)$.

The Bethe free energy approximates $H(b)$ by $\sum_\alpha H(b_\alpha) + \sum_i (1 - d_i) H(b_i)$ where d_i is the degree of the variable node i, $H(b_\alpha) = -\sum_{\mathbf{x}_\alpha} b_\alpha(\mathbf{x}_\alpha) \ln b_\alpha(\mathbf{x}_\alpha)$ and $H(b_i) = -\sum_{x_i} b(x_i) \ln b(x_i)$. As a result, the computational complexity of the Bethe free energy is exponential only in the size of the cliques. The Bethe free energy is exact (equal to free energy) when the factor graph has no cycles and in that case the energy is strictly convex over the set of constraints mentioned above. When the factor graph has cycles the Bethe free energy is non-convex. Another notable property is that the fixed-points of the BP algorithm correspond to local minima of the Bethe free energy minimization over the constraints on b (Yedidia et al., 2005).

The Bethe approximation of the entropy $H(b)$ can be

written in a more general form as:

$$\sum_{\alpha} \bar{c}_{\alpha} H(b_{\alpha}) + \sum_{i} \bar{c}_{i} H(b_{i}), \tag{1}$$

where $\bar{c}_i = 1 - \sum_{\alpha \in N(i)} \bar{c}_{\alpha}$. Thus when the coefficients $\bar{c}_{\alpha} = 1$ for all factor nodes we obtain the Bethe approximation. A *convex free energy* is based on a result of (Heskes, 2004) who derived sufficient conditions for an entropy approximation to be convex over the set of constraints. In the setting we have described, those conditions have the following form (Weiss et al., 2007):

Definition: An approximate entropy term of the form eqn. 1 is strictly convex over the set of constraints $\sum_{\mathbf{x}_{\alpha} \setminus x_i} b_{\alpha}(\mathbf{x}_{\alpha}) = b_i(x_i)$ for all $\alpha \in N(i)$ if there exists $c_i, c_{i\alpha} \geq 0$ and $c_{\alpha} > 0$ such that $\bar{c}_{\alpha} = c_{\alpha} + \sum_{i \in N(\alpha)} c_{i\alpha}$ and $\bar{c}_i = c_i - \sum_{\alpha \in N(i)} c_{i\alpha}$. The approximate entropy becomes:

$$\sum_{i,\alpha \in N(i)} c_{i\alpha}(H(b_{\alpha}) - H(b_i)) + \sum_{\alpha} c_{\alpha} H(b_{\alpha}) + \sum_{i} c_i H(b_i)$$

Taken together, the convex free energy constrained optimization problem is:

$$\begin{aligned} \min_{b_{\alpha}, b_i} \quad & \sum_{\alpha} \sum_{\mathbf{x}_{\alpha}} E_{\alpha}(\mathbf{x}_{\alpha}) b_{\alpha}(\mathbf{x}_{\alpha}) - \sum_{\alpha} c_{\alpha} H(b_{\alpha}) \\ & - \sum_{i} c_i H(b_i) + \sum_{i,\alpha \in N(i)} c_{i\alpha}(H(b_i) - H(b_{\alpha})) \\ & \text{subject to} \\ & \sum_{\mathbf{x}_{\alpha} \setminus x_i} b_{\alpha}(\mathbf{x}_{\alpha}) = b_i(x_i) \quad \forall \alpha \in N(i) \\ & \sum_{\mathbf{x}_{\alpha}} b_{\alpha}(\mathbf{x}_{\alpha}) = 1 \quad \forall \alpha \\ & b_{\alpha}, b_i \geq 0 \end{aligned} \tag{2}$$

We denote the criterion function (convex free energy) by $F_{con}(b_{\alpha}, b_i)$ and note that it is strictly convex over the set of constraints provided that $c_i, c_{i\alpha} \geq 0$ and $c_{\alpha} > 0$. For now we assume that the parameters $c_i, c_{i\alpha}, c_{\alpha}$ are given as input and set out to derive a message-passing algorithm (two versions, one sequential and the other parallel) which is guaranteed to converge to the global minimum for any factor graph. Later in Section 5 we will introduce an algorithm for determining the convex free energy parameters from the factor graph.

3 A General Framework for Sequential and Parallel Message Passing Algorithms

The constrained minimization of eqn. 2 can be handled within the body of convex programming tools. Those, however, have a high computational cost and their architecture (the update flow of parameters) is far from similar to a message passing architecture and to BP in particular. In this section we will take a detour and develop a message passing framework (sequential and parallel versions) to a particular sub-class of convex problems. Later in Section 4 we will map the convex free energy minimization of eqn. 2 into this framework. Consider the class of problems

$$\min_{b} f(b) + \sum_{i=1}^{n} h_i(b),$$

where $b \in R^m$, $f(b)$ is a strictly convex real valued function and $h_i(b)$ are convex (not necessarily strict nor differentiable), and *proper* (i.e., can take the value ∞ for some values of b). This class of problems includes in particular the classical convex program: $\min_b f(b)$ over the constraints $b \in C_1 \cap \cdots \cap C_n$ where C_i are convex sets[1].

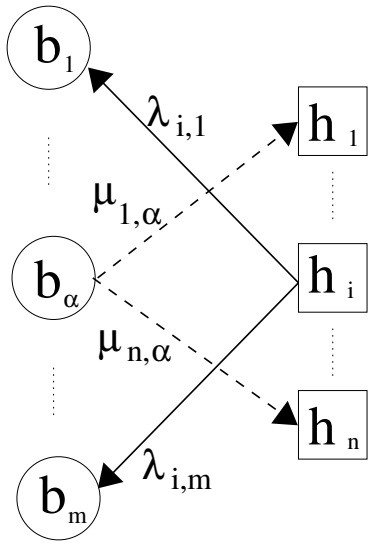


Figure 1: *Message-passing architecture of Algorithms 1,2*

For this class of problems we derive two message-passing algorithms, one sequential and the other parallel. Both algorithms are based on a block update regime using convex duality. The sequential algorithm is described below:

Algorithm 1 (Sequential Message-Passing) *Let* $\boldsymbol{\lambda}_i$ *and* $\boldsymbol{\mu}_i$, $i = 1, ..., n$ *be vectors in* R^m. *Set* $\boldsymbol{\lambda}_i = 0$.

1. *For* $t = 1, 2, ...$
2. *For* $i = 1, ...n$:
   (a) $\boldsymbol{\mu}_i \leftarrow \sum_{j \neq i} \boldsymbol{\lambda}_j$

[1] In this case $h_i(b) = \delta_{C_i}(b)$ is the indicator function $\delta_{C_i}(b) = 0$ if $b \in C_i$ and $\delta_{C_i}(b) = \infty$ if $b \notin C_i$.

(b) $\mathbf{b}^* \leftarrow \underset{\mathbf{b}\in domain(h_i)}{\operatorname{argmin}} \Big\{h_i(\mathbf{b}) + f(\mathbf{b}) + \mathbf{b}^\top \boldsymbol{\mu}_i\Big\}$.

(c) $\boldsymbol{\lambda}_i \leftarrow -\boldsymbol{\mu}_i - \nabla f(\mathbf{b}^*)$.

Output $\mathbf{b}^*$.

The vectors $\boldsymbol{\lambda}_i$ and $\boldsymbol{\mu}_i$ are messages passed along edges of a bipartite graph with n (function) nodes corresponding to the n functions $h_i(b)$ and m (variable) nodes corresponding to the dimension of b. Function node i sends the m coordinates of vector $\boldsymbol{\lambda}_i$ to the m variable nodes. Variable node j sends the j'th coordinate of vectors $\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_n$ to the n functions nodes (see Fig. 1). The algorithm is "sequential" in the sense that it is crucial to move sequentially over the index $i = 1, ..., n$, thus the network proceeds in a node-after-node update policy.

For those familiar with successive projection schemes, in the particular case when $h_i(b) = \delta_{C_i}(b)$ (the indicator function of convex set C_i), the update step for b is a "Bregman" projection (Bregman et al., 1999) of the vector $\boldsymbol{\mu}_i$ onto the convex set C_i. In that case, following some algebraic manipulations (such as eliminating $\boldsymbol{\mu}_i$ among other manipulations) the scheme reduces to the well known Dykstra (Dykstra, 1983) (also goes under different names such as Hildreth, Bregman, Csiszar, Han, Tseng) successive projection algorithm which has its origins in the work of Von-Neumann (von Neumann, 1950). We introduce next a *parallel* message-passing algorithm:

Algorithm 2 (Parallel Message-Passing) *Let* $\boldsymbol{\lambda}_i$ *and* $\boldsymbol{\mu}_i$, $i = 1, ..., n$ *be vectors in* R^m. *Set* $\boldsymbol{\mu}_i = 0$.

1. *For* $t = 1, 2, ...$
2. *For* $i = 1, ...n$ *in parallel*

$$\mathbf{b}^* \leftarrow \underset{\mathbf{b}\in domain(h_i)}{\operatorname{argmin}} \left\{\frac{1}{n} f(\mathbf{b}) + h_i(\mathbf{b}) + \mathbf{b}^\top \boldsymbol{\mu}_i\right\}$$

$$\boldsymbol{\lambda}_i \leftarrow -\boldsymbol{\mu}_i - \frac{1}{n}\nabla f(\mathbf{b}^*)$$

3. *For* $i = 1, ...n$ *in parallel*

$$\boldsymbol{\mu}_i \leftarrow -\boldsymbol{\lambda}_i + \frac{1}{n}\sum_{j=1}^{k} \boldsymbol{\lambda}_j$$

Output $\mathbf{b}^*$.

The description of the messages and the network architecture are the same as in the sequential algorithm but here all the function nodes update and send their messages in parallel to the variable nodes. Once all the variable nodes have received their messages they compute their update and send their message in parallel to the function nodes. The derivation of both algorithms is presented in the Appendix.

4 Convergent Message-Passing Algorithms for Convex Free Energies

We are ready to derive a convergent algorithm for the constrained convex free energy minimization problem (eqn. 2) using the two algorithms above. It is important to note that in the framework of $f(b) + \sum_i h_i(b)$ the function $f(b)$ is strictly convex in the entire domain whereas the convex free energy $F_{con}(b_\alpha, b_i)$ is strictly convex *over the set of constraints*. We need therefore both to map $F_{con}(b_\alpha, b_i)$ onto the framework of $f(b) + \sum_i h_i(b)$ and handle the convexity over the domain of definition issue.

Let $b_\alpha(x_i)$ stand for $\sum_{\mathbf{x}_\alpha \setminus x_i} b_\alpha(\mathbf{x}_\alpha)$ (a notation also used by (Heskes, 2006)). The marginal constraints dictate that $b_\alpha(x_i) = b_\beta(x_i)$ for all $\alpha, \beta \in N(i)$. We substitute $b_\alpha(x_i)$ for $b_i(x_i)$ in eqn. 2 and move terms around to fit the $f(b)+\sum_i h_i(b)$ framework. The result is summarized below:

Let $b = (b_{\alpha_1}, ..., b_{\alpha_m})$, that is we dropped $b_i(x_i)$ from the process. Let $f(b)$ be defined as follows:

$$\begin{aligned} f(b) &= \sum_\alpha \left(\sum_{x_\alpha} E_\alpha(x_\alpha) b_\alpha(x_\alpha) - c_\alpha H_\alpha(b_\alpha)\right) \quad (3) \\ &= \sum_\alpha f_\alpha(b_\alpha) \end{aligned}$$

Let $h_i(b)$, $i = 1, ..., n$, corresponding to the n variable nodes of the factor graph, be defined below:

$$h_i(\mathbf{b}) = \left\{ \begin{array}{ll} \infty & \exists \alpha, \beta \in N(i) : b_\alpha(x_i) \neq b_\beta(x_i) \\ \sum_{\alpha\in N(i)} h_{i\alpha}(b_\alpha) & otherwise \end{array} \right\} \quad (4)$$

where $h_{i\alpha}(b_\alpha)$ is defined as follows:

$$h_{i\alpha}(b_\alpha) = (c_i/|N(i)| - c_{i\alpha}) \sum_{x_i} b_\alpha(x_i) \ln b_\alpha(x_i) - c_{i\alpha} H(b_\alpha) \quad (5)$$

Taken together, the constrained convex free-energy minimization (eqn. 2) becomes:

$$\min_{b=(b_{\alpha_1},...,b_{\alpha_m})} f(b) + \sum_{i=1}^{n} h_i(b) \quad s.t. \quad b \geq 0, \sum_{x_\alpha} b_\alpha(x_\alpha) = 1,$$

where now $f(b)$ is strictly convex over its entire domain of definition and h_i are proper convex functions. Note that $f(b)$ and $h_i(b)$ each decompose into a sum of simpler functions, i.e., $f(b) = \sum_\alpha f_\alpha(b_\alpha)$ and $h_i(b) = \sum_{\alpha\in N(i)} h_{i\alpha}(b_\alpha)$. The decomposition translates into the messages being sparse. For example, the update step of b^* in both message-passing algorithms becomes:

$$b^*_{i,\alpha\in N(i)} = \underset{\mathbf{b}\in dom(h_i)}{\operatorname{argmin}} \sum_{\alpha\in N(i)} (f_\alpha(b_\alpha) + h_{i\alpha}(b_\alpha) + b_\alpha^\top \boldsymbol{\mu}_{i\alpha})$$

1. Set $n_{j\to\gamma}(x_\gamma) = 1$ for all $i = 1, ..., n$, $\gamma \in N(i)$ and $\mathbf{x}_\gamma$.

2. For $t = 1, 2, ...$

3. For $i = 1, ...n$:

$$
\begin{aligned}
m_{\gamma\to i}(x_i) &= \sum_{z_\gamma \setminus x_i} \left(\psi_\gamma(z_\gamma) \prod_{j\in N(\gamma)\setminus i} n_{j\to\gamma}(z_\gamma) \right)^{1/\hat{c}_{i\gamma}} \\
b^*_\gamma(x_i) &\propto \prod_{\alpha\in N(i)} m^{\hat{c}_{i\alpha}/\hat{c}_i}_{\alpha\to i}(x_i), \\
n_{i\to\gamma}(x_\gamma) &= \left(\psi_\gamma(x_\gamma) \prod_{j\in N(\gamma)\setminus i} n_{j\to\gamma}(x_\gamma) \right)^{-\frac{c_{i\gamma}}{\hat{c}_{i\gamma}}} \left(\frac{b^*_\gamma(x_i)}{m_{\gamma\to i}(x_i)} \right)^{c_\gamma}
\end{aligned}
$$

Figure 2: Sequential message-passing algorithm for convex free energy. The constants $\hat{c}_i$ and $\hat{c}_{i\alpha}$ are defined as $\hat{c}_i = c_i + \sum_{\alpha\in N(i)} c_\alpha$ and $\hat{c}_{i\alpha} = c_\alpha + c_{i\alpha}$.

where $b_{i,\alpha\in N(i)}$ are the entries b_α in b corresponding to factor nodes α neighboring to variable node i. Likewise $\boldsymbol{\mu}_{i\alpha}$ are the portions in $\boldsymbol{\mu}_i$ corresponding to factor nodes neighboring to variable node i. The domain of h_i are the constraints :

$$
\begin{aligned}
&\sum_{\mathbf{x}_\alpha} b_\alpha(\mathbf{x}_\alpha) = 1, \quad \forall\alpha \in N(i) \\
&b_\alpha(x_i) = b_\beta(x_i), \quad \forall\alpha, \beta \in N(i)
\end{aligned}
$$

We omit the remainder of the derivation as it is long, tedious and mechanical and arrive to the final description of the two message-passing algorithms presented in Fig. 2 and Fig. 3. Like BP, the algorithms send messages between variable nodes and factor nodes where $n_{i\to\gamma}(x_\gamma)$ represents the message from variable node i to factor node γ and $m_{\gamma\to i}(x_\gamma)$ is the message from factor node γ to variable node i.

5 Fitting a Convex Free Energy to a Graphical Model

The convex free energy contains three sets of parameters $c_i, c_{i\alpha} \geq 0$ and $c_\alpha > 0$ one parameter per each node and edge in the factor graph. Our discussion so far was general in the sense that we presented algorithms for handling the family of convex free energies without regard as to how those parameters are determined. The only setting of parameters proposed to date is the tree-reweighted (TRW) free energy where $\bar{c}_\alpha$ can be set analytically. This has a simple form when the cliques are of size 2, i.e., representing pairs of variables. In that case, $\bar{c}_\alpha$ is the number of spanning trees containing α divided by the total number of spanning trees (a computation which can be done analytically). Once $\bar{c}_\alpha$ is determined then $\bar{c}_i = 1 - \sum_{\alpha\in N(i)} \bar{c}_\alpha$ and from $\bar{c}_\alpha$ and $\bar{c}_i$ one can solve for $c_i, c_{i\alpha} \geq 0$ and $c_\alpha > 0$ by means of linear satisfaction from the equations:

$$
\bar{c}_\alpha = c_\alpha + \sum_{i\in N(\alpha} c_{i\alpha}, \quad \bar{c}_i = c_i - \sum_{\alpha\in N(i)} c_{i\alpha}. \tag{6}
$$

In this section we propose a heuristic for setting the parameters based on the following idea[2]. Given the equations (eqn. 6) connecting the parameters to $\bar{c}_\alpha$ and $\bar{c}_i$, the space of admissible solutions must satisfy the following equations:

$$
\begin{aligned}
&c_i + \sum_{\alpha\in N(i)} (c_\alpha + \sum_{j\in N(\alpha)\setminus i} c_{j\alpha}) = 1, \quad i = 1, ..., n \\
&c_i, c_{i\alpha} \geq 0, \quad c_\alpha > 0.
\end{aligned}
$$

Among all possible admissible solutions we choose the one in which $\bar{c}_\alpha$ is as uniform as possible, i.e., we apply Laplace's principle of insufficient reasoning. The criterion function, therefore, minimizes:

$$
\min_{c_i, c_{i\alpha}, c_\alpha \in admissible} \sum_\alpha (c_\alpha + \sum_{i\in N(\alpha)} c_{i\alpha} - 1)^2, \tag{7}
$$

which is a least-squares criteria for uniformity of $\bar{c}_\alpha$. Alternatively, we also used the maximum entropy approach where the criterion function minimizes $\sum_\alpha \bar{c}_\alpha \ln \bar{c}_\alpha$. In both cases we used standard solvers to recover $c_i, c_{i\alpha}, c_\alpha$, i.e., we did not attempt to devise specifically tailored solvers for those problems.

The desire towards uniformity, besides being used extensively in probabilistic settings, is motivated by the success of the Bethe free energy where $\bar{c}_\alpha = 1$. The Bethe free energy is non-convex for factor graphs with

[2] A similar idea was independently derived by Nir Friedman and his collaborators — personal communication.

cycles, thus is not a member of the convex free energies, but empirical evidence suggest that when BP converges the marginals are surprisingly good. For Bethe free energy $\bar{c}_\alpha = 1$ over all factor nodes α — hence our proposal to strive for uniformity over the space of admissible solutions. In some sense we are attempting to "convexify" the Bethe free energy, although this is not being done directly.

6 Experiments

We applied our (parallel and sequential) message passing algorithm using the heuristic (eqn. 7) for setting the parameters of the convex free energy from the input graph to an Ising model on a two dimensional 8×8 grid. The distribution has the form $p(\mathbf{x}) \propto e^{\sum_{ij \in E} \theta_{ij} x_i x_j + \theta_i x_i}$, where θ_{ij}, θ_i are parameters, $x_i \in \{\pm 1\}$, and E are edges of the 2D grid.

Following (Globerson & Jaakkola, 2007a), the parameters θ_i were drawn uniformly from $\mathcal{U}[-d_f, d_f]$ where $d_f \in \{0.05, 1\}$. The parameters θ_{ij} were drawn from $\mathcal{U}[-d_o, d_o]$ or $\mathcal{U}[0, d_o]$ to obtain *mixed* or *attractive* interaction potential respectively. The interaction levels were $d_o \in \{0.2, 0.4, ..., 4\}$. In addition to BP[3], the following algorithms were used to estimate the marginals of the distribution:

- CCCP algorithm (Yuille, 2002) for obtaining a local minima of the constrained Bethe free energy. Our message-passing algorithm runs on a "covexified" version of the Bethe free-energy and achieves its global minimum. The CCCP, on the other hand, runs on the (non-convex) Bethe energy and finds a local minima.
- Sequential MP where the convex free energy parameters determined using the convex-L_2 free energy described in Eqn.7.
- The same as above but the parameters were set using maximum entropy (instead of L_2), we call convex-H.
- The TRW method of (Wainwright et al., 2005) with uniform distributions over the trees. Since the TRW free energy belongs to the class of convex free energies we ran our sequential MP algorithm on the parameters $c_i, c_{i\alpha}, c_\alpha$ determined by TRW.

For each setting of the parameters and each algorithm we calculated the mean L_1 error in the marginals $\frac{1}{n}\sum_i |p^{(alg)}(x_i = 1) - p^{(true)}(x_i = 1)|$. The accuracy

[3]We used the inference package by Talya Meltzer available at http://www.cs.huji.ac.il/~talyam/.

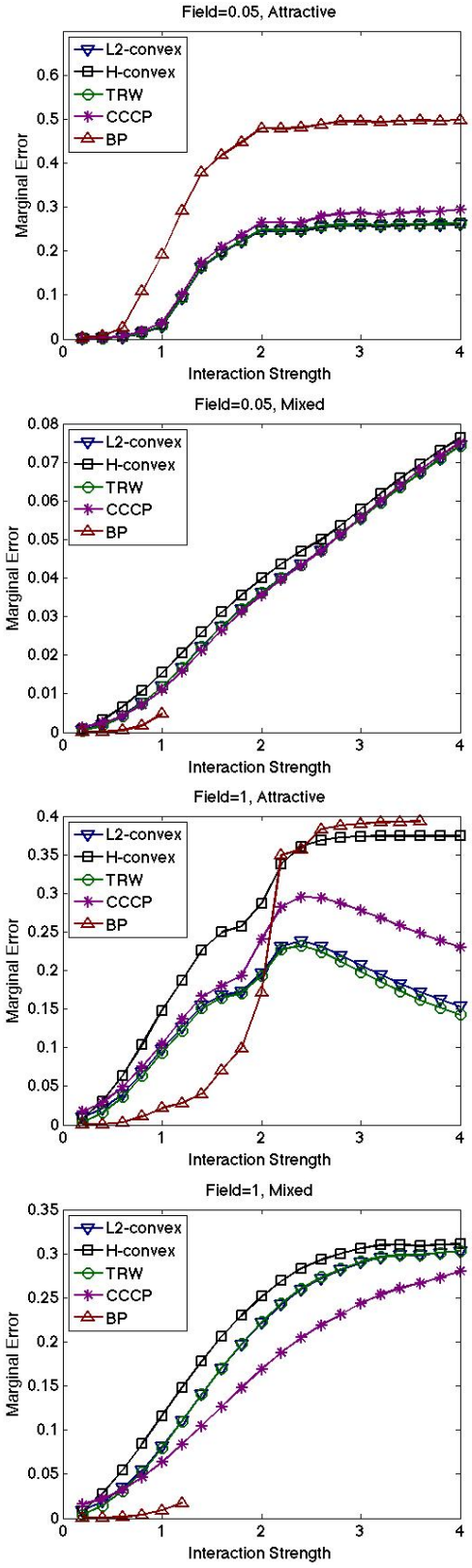


Figure 4: *Comparison of error in marginals estimation on an Ising model on a two dimensional 8×8 grid. The models presented include BP (when converged), TRW, CCCP, convex-L_2 and convex-H. Mean is shown for 10 random trials.*

1. For $t = 1, 2, ...$
2. For $i = 1, ...n$ in parallel

$$m_{\gamma \to i}(x_i) = \sum_{z_\gamma \setminus x_i} \left(\psi_\gamma(z_\gamma) \frac{\prod_{j \in N(\gamma)} n_{j \to \gamma}(z_\gamma)^{1/n}}{n_{i \to \gamma}(z_\gamma)} \right)^{1/\hat{c}_{i\gamma}}$$

3. For $i = 1, ...n$ in parallel
   (a) For every x_i:

$$b^*_\gamma(x_i) \propto \prod_{\alpha \in N(i)} m^{\hat{c}_{i\alpha}/\hat{c}_i}_{\alpha \to i}(x_i)$$

   for every $\gamma \in N(i)$ and every $x_\gamma \setminus x_i$:

$$n_{i \to \gamma}(x_\gamma) = \frac{n_{i \to \gamma}(x_\gamma)}{(\psi_\gamma(x_\gamma) \prod_{j \in N(\gamma)} n_{j \to \gamma}(x_\gamma))^{1/n}} \left(\psi_\gamma(z_\gamma) \frac{\prod_{j \in N(\gamma)} n_{j \to \gamma}(x_\gamma)^{1/n}}{n_{i \to \gamma}(x_\gamma)} \right)^{c_\gamma / n \hat{c}_{i\gamma}} \left(\frac{b^*_\gamma(x_i)}{m_{\gamma \to i}(x_i)} \right)^{c_\gamma / n}$$

Figure 3: Parallel message-passing algorithm for convex free energy. The constants $\hat{c}_i$ and $\hat{c}_{i\alpha}$ are defined as $\hat{c}_i = c_i + \sum_{\alpha \in N(i)} c_\alpha$ and $\hat{c}_{i\alpha} = c_\alpha + c_{i\alpha}$.

results are shown in Fig. 4. The displays are arranged into four cases: Field=0.05, 1 and Mixed versus Attractive interactions. In three out of the four cases, the performance of the three convex free energies models are roughly the same. In the case [Field=1, Attractive], TRW and convex-L_2 produce roughly the same marginal approximations and convex-H is worse. BP does not converge in the Mixed cases for high Interaction value of d_o; CCCP produces comparable results in the two cases when Field=0.05, and produces the best results for [Field=1, Mixed]. To conclude so far, the two convex free energy settings produce comparable results to TRW on all cases and are sometimes better and sometimes worse than BP (when converges). Among the two setting of the convex free energy, convex-L_2 consistently produces better approximations than convex-H.

It is interesting that CCCP often produces good approximations, however this comes at a costly run-time tradeoff. Fig. 5 compares the running time of our (sequential) MP algorithm with a general convex solver performing conditional gradient descent on the primal energy function (Bertsekas et al., 2003) which uses linear programming to find feasible search directions, and to the CCCP algorithm. We ran all three algorithms on $n \times n$ grids where $n = 2, 3, ..., 10$. The stopping criteria for all algorithms was the same and based on a primal energy difference of 10^{-5}. For a 10×10 grid, for instance, the general convex solver was slower by a factor of 20 (e.g., 306 seconds compared to 15.2) and the CCCP was slower by a factor of 115 compared to our MP algorithm (running 1767 seconds). For a 2×2 grid, on the other hand, our MP algorithm took 0.15 seconds compared to 0.59 for CCCP and 1.41 seconds for the general convex solver.

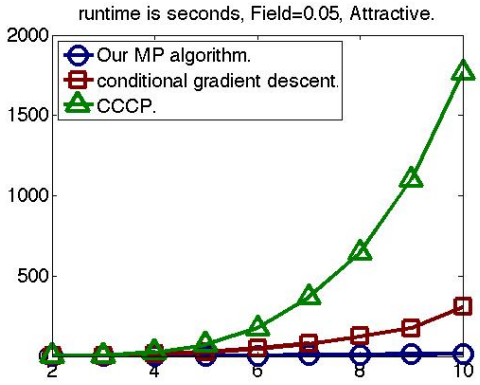


Figure 5: *Run-time (in seconds) comparisons of our message-passing algorithm against a conditional gradient descent solver (running on convex-L_2 free energy) and against CCCP for the (non-convex) Bethe energy. All three algorithms were applied to $n \times n$ grids with $n = 2, 3..., 10$. Mean is shown for 10 random trials.*

Our next experiment is conducted on random graphs to analyze the differences between BP, CCCP, TRW, convex-H and convex-L_2. To generate a random graph we used the probability space of $G(n, p)$ over graphs with n vertices - where each edge is present with probability p and absent with probability $1 - p$, independently among edges. Note that in $G(n, \frac{1}{2})$ all the graphs have the same probability, i.e., $G(n, \frac{1}{2})$ is the probability space consisting of all n-vertex graphs under the uniform distribution.

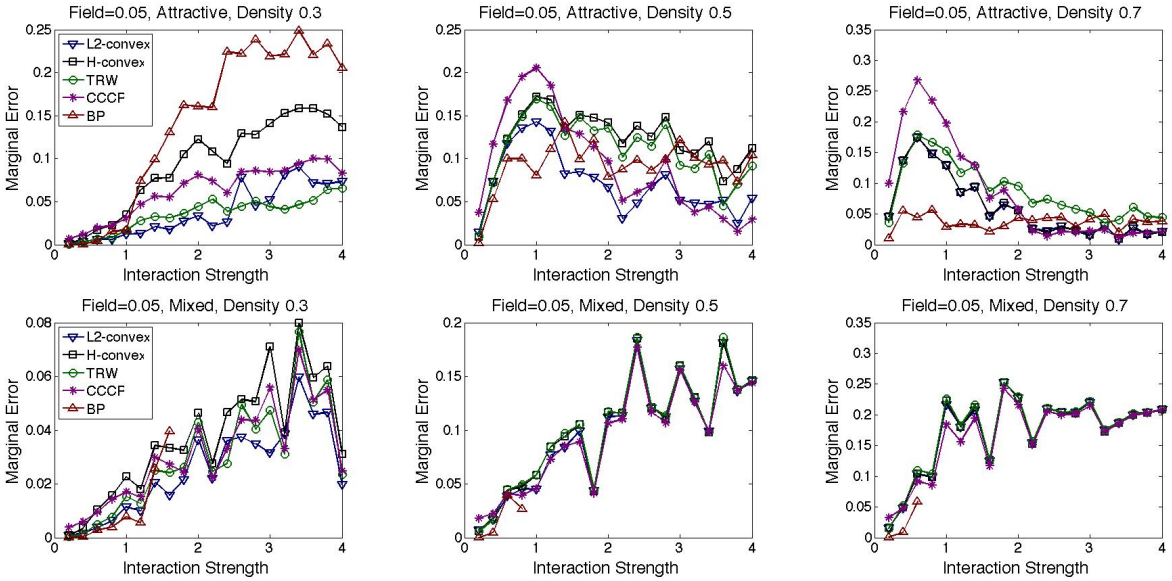


Figure 6: *Comparison of error in marginals estimation for random graphs with edge density* $p = 0.3, 0.5, 0.7$ *(left to right) and for local field value of* 0.05*. For high field value* ($d_f = 1$*, not displayed here) convex-*L_2*, convex-H and TRW produce similar results. Mean is shown for* 10 *random trials.*

In Fig. 6 we observe that the typical graph behavior is mainly dependent on the edge-density of the random graph as well as on the interaction levels. In order to compute the exact marginals we chose 10 vertices. A random graph with edge density of $p = 0.3$ is "almost" a tree and we see that the TRW is slightly inferior to convex-L_2 and better than convex-H. For intermediate edge-density $p = 0.5$ the TRW and convex-H are comparable and both are inferior to convex-L_2. When edge-density is high, i.e., $p = 0.7$ the graph is far from a tree and the convex-L_2 as well as convex-H are better than TRW. Note that in the Mixed case all the convex algorithms produce comparable results. In those cases BP usually does not converge. Nevertheless, in other cases (except $p = 0.3$, Attractive) BP produces good marginal approximation (consistent with empirical observations found in the literature) and that convex-L_2 is not far behind. It is interesting to note that, unlike the 8×8 grid, the CCCP is in most situations inferior to the convex free energy models.

7 Summary and Discussion

The convex free energies provide a way for obtaining approximate inference over general graphs. There are two main issues in this regard: the first is how to obtain a guaranteed globally convergent message-passing algorithm for the general class of convex free energies, and secondly, how to tune the energy parameters $c_i, c_{i\alpha}, c_\alpha$ to a specific graph?

As for the first issue, we have provided a complete treatment by deriving both sequential and parallel convergent message-passing algorithms which have similar form to BP. The algorithms are based on a general message-passing architecture designed for a class of problems of the type $f(b) + \sum_i h_i(b)$ with $f(b)$ being strictly convex and h_i being convex, continuous and proper. We have shown the basic steps of fitting the constrained convex free energy problem into this framework. We limited the discussions to factor graphs where the neighborhoods of every pair of factor nodes have at most a single intersection. This limitation can easily be removed by replacing the term $c_{i\alpha}(H(b_\alpha) - H(b_i))$ with the term $c_{\alpha,\beta}(H(b_\alpha) - H(b_\beta))$ for every pair of factor nodes. This replacement propagates mechanically into subsequent steps of the derivation — as would be found in a more detailed follow-up of this paper.

As for the second issue, we have proposed a heuristic principle where among all admissible parameters we choose the one most closest to the Bethe free energy (using Laplace principle of insufficient reasoning). Empirical results show that for certain graphs, like a grid, we obtain very close marginal results to those obtained by the TRW free energy. For random graphs we obtain a very different free energy from TRW and superior accuracy of marginal estimation. The results suggest that our heuristic for setting up the convex free energy satisfies what we were after, i.e., to get approximations similar to BP but in guaranteed (globally) convergent framework. Future work is required for obtaining a firmer theoretical understanding about the applicability of our heuristic and relation to TRW

free energy in particular.

References

Bertsekas, D., Nedić, A., & Ozdaglar, A. (2003). *Convex analysis and optimization*. Athena Scientific Belmont, Mass.

Bregman, L., Censor, Y., & Reich, S. (1999). Dykstras algorithm as the nonlinear extension of Bregmans optimization method. *Journal of Convex Analysis*, *6*, 319–333.

Dykstra, R. (1983). An Algorithm for Restricted Least Squares Regression. *Journal of the American Statistical Association*, *78*, 837–842.

Globerson, A., & Jaakkola, T. (2007a). Approximate inference using conditional entropy decompositions.

Globerson, A., & Jaakkola, T. (2007b). Convergent Propagation Algorithms via Oriented Trees. *Uncertainty in Artificial Intelligence (UAI 2007)*.

Heskes, T. (2004). On the Uniqueness of Loopy Belief Propagation Fixed Points. *Neural Computation*, *16*, 2379–2413.

Heskes, T. (2006). Convexity Arguments for Efficient Minimization of the Bethe and Kikuchi Free Energies. *Journal of Artificial Intelligence Research*, *26*, 153–190.

Kschischang, F., Frey, B., & Loeliger, H. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*, 498–519.

Mooij, J.M., & Kappen, H.J. (2005). Sufficient conditions for convergence of loopy belief propagation. *Uncertainty in Artificial Intelligence (UAI 2005)*.

Murphy, K., Weiss, Y., & Jordan, M. (1999). Loopy belief propagation for approximate inference: An empirical study. *Proceedings of Uncertainty in AI*, 467–475.

Rockafellar, R. (1970). *Convex Analysis*. Princeton University Press.

Tseng, P. (1993). Dual coordinate ascent methods for non-strictly convex minimization. *Mathematical Programming*, *59*, 231–247.

von Neumann, J. (1950). Functional Operators, Vol. II: The Geometry of Orthogonal Spaces. *Annals of Math. Studies*, *22*.

Wainwright, M., Jaakkola, T., & Willsky, A. (2005). A new class of upper bounds on the log partition function. *Information Theory, IEEE Transactions on*, *51*, 2313–2335.

Weiss, Y., Yanover, C., & Meltzer, T. (2007). MAP Estimation, Linear Programming and Belief Propagation with Convex Free Energies. *Uncertainty in Artificial Intelligence (UAI 2007)*.

Yedidia, J., Freeman, W., & Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, *51*, 2282–2312.

Yuille, A. (2002). CCCP Algorithms to Minimize the Bethe and Kikuchi Free Energies: Convergent Alternatives to Belief Propagation. *Neural Computation*, *14*, 1691–1722.

A Sequential and Parallel Block Updates for $\min_b f(b) + \sum_i h_i(b)$

Recall the $f(b)$ is a strictly convex real-valued function and the functions h_i are convex, proper and continuous. We quote below two basic theorems from convex duality (cf. (Bertsekas et al., 2003)) which we will use as building blocks for our algorithms.

Theorem 1 Basic Fenchel Duality I
Let $g(\mathbf{b})$ be a convex and differentiable function and let $h(\mathbf{b})$ be a proper convex and continuous function, and let $h^*(\boldsymbol{\lambda}) = \max_{\mathbf{b}} \left\{ \mathbf{b}^\top \boldsymbol{\lambda} - h(\mathbf{b}) \right\}$ be its conjugate dual function. Consider the primal and dual programs:

$$\begin{aligned} \textit{Primal:} \quad & \min_{\mathbf{b}} g(\mathbf{b}) + h(\mathbf{b}) \\ \textit{Dual:} \quad & \max_{\boldsymbol{\lambda}} \left\{ \min_{\mathbf{b}} \left(g(\mathbf{b}) + \mathbf{b}^\top \boldsymbol{\lambda} \right) - h^*(\boldsymbol{\lambda}) \right\} \end{aligned}$$

then there is no duality gap and the optimal primal-dual pair $\mathbf{b}^*, \boldsymbol{\lambda}^*$ satisfies $\nabla g(\mathbf{b}^*) = -\boldsymbol{\lambda}^*$.

The next theorem is a generalized version of the one above:

Theorem 2 Basic Fenchel Duality II
Let $f(\mathbf{b})$ be a strictly convex and differentiable function and let $h_i(\mathbf{b})$ be proper convex and continuous functions, and let $h_i^*(\boldsymbol{\lambda}) = \max_{\mathbf{b}} \left\{ \mathbf{b}^\top \boldsymbol{\lambda} - h_i(\mathbf{b}) \right\}$ be their conjugate dual functions. Consider the primal and dual programs:

$$\begin{aligned} \textit{Primal:} \quad & \min_{\mathbf{b}} f(\mathbf{b}) + \sum_{i=1}^{n} h_i(\mathbf{b}) \\ \textit{Dual:} \quad & \max_{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_n} \left\{ \min_{\mathbf{b}} \left(f(\mathbf{b}) + \mathbf{b}^\top \sum_{i=1}^{n} \boldsymbol{\lambda}_i \right) - \sum_{i=1}^{n} h_i^*(\boldsymbol{\lambda}_i) \right\} \end{aligned}$$

then there is no duality gap, and the optimal primal-dual pair $\mathbf{b}^*, \boldsymbol{\lambda}_i^*$ satisfies $\nabla f(\mathbf{b}^*) = -\sum_{i=1}^{n} \boldsymbol{\lambda}_i^*$.

A.1 The Sequential Block Update Algorithm

Since $f(\mathbf{b})$ is strictly convex, then its conjugate dual $\min_{\mathbf{b}} \left(f(\mathbf{b}) + \mathbf{b}^\top \sum_{i=1}^{n} \boldsymbol{\lambda}_i \right)$ is differentiable (see (Rockafellar, 1970)). In this case a block dual ascent optimization scheme converges to the global maxima (Tseng, 1993). Our algorithm alternates over $\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_n$ by optimizing $\boldsymbol{\lambda}_i$ while fixing $\boldsymbol{\lambda}_j$ for $j \neq i$.

Let $\boldsymbol{\mu}_i = \sum_{j \neq i} \boldsymbol{\lambda}_i$ and define the following dual algorithmic building block:

$$\max_{\boldsymbol{\lambda}_i} \left\{ \min_{\mathbf{b}} \left(f(\mathbf{b}) + \mathbf{b}^\top \boldsymbol{\mu}_i + \mathbf{b}^\top \boldsymbol{\lambda}_i \right) - h_i^*(\boldsymbol{\lambda}_i) \right\} \quad (8)$$

To recover $\boldsymbol{\lambda}_i$ one can use Theorem 1: Set $g(\mathbf{b}) \leftarrow f(\mathbf{b}) + \mathbf{b}^\top \boldsymbol{\mu}_i$ and $h(\mathbf{b}) \leftarrow h_i(\mathbf{b})$, and solve the primal program:

$$\mathbf{b}^* = \operatorname*{argmin}_{\mathbf{b} \in domain(h_i)} \left\{ f(\mathbf{b}) + \mathbf{b}^\top \boldsymbol{\mu}_i + h_i(\mathbf{b}) \right\}$$

From the Lagrange optimality condition of Theorem 1 we recover $\boldsymbol{\lambda}_i^*$

$$\boldsymbol{\lambda}_i^* = -\boldsymbol{\mu}_i - \nabla f(\mathbf{b}^*)$$

Taken together, one obtains Algorithm 1 described in Section 3.

A.2 The Parallel Block Update Algorithm

We begin by stating and proving the following theorem:

Theorem 3 *Let $f(\mathbf{b})$ be a strictly convex and differentiable function and let $h_i(\mathbf{b})$ be proper convex and continuous functions, and let $h_i^*(\boldsymbol{\lambda}) = \max_{\mathbf{b}} \left\{ \mathbf{b}^\top \boldsymbol{\lambda} - h_i(\mathbf{b}) \right\}$ be their conjugate dual functions. The following is a primal/dual pair with no duality-gap:*

$$(P) \min_{\mathbf{b}} f(\mathbf{b}) + \sum_{i=1}^n h_i(\mathbf{b})$$

$$(D) \max_{\substack{\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_n \\ \sum_{i=1}^n \boldsymbol{\mu}_i = 0}} \left\{ \sum_{i=1}^n \left(\min_{\mathbf{b}} \left(\frac{1}{n} f(\mathbf{b}) + \mathbf{b}^\top (\boldsymbol{\lambda}_i + \boldsymbol{\mu}_i) \right) - h_i^*(\boldsymbol{\lambda}_i) \right) \right\}$$

Furthermore, the optimal primal-dual pair $\mathbf{b}^*, \boldsymbol{\lambda}_i^*$ satisfies $\nabla f(\mathbf{b}^*) = -\sum_{i=1}^n \boldsymbol{\lambda}_i^*$.

Proof: we introduce an equivalent primal function $\sum_{i=1}^n \left(\frac{1}{n} f(\mathbf{b}_i) + h_i(\mathbf{y}_i) \right)$ subject to the constraints $\mathbf{b} = \mathbf{b}_i$ and $\mathbf{b}_i = \mathbf{y}_i$ for every i. The Lagrangian $L(\mathbf{b}, \mathbf{y}_i, \boldsymbol{\lambda}_i, \boldsymbol{\mu}_i)$ and the Lagrange dual function $q(\boldsymbol{\lambda}_i, \boldsymbol{\mu}_i) = \min_{\mathbf{b}, \mathbf{b}_i, \mathbf{y}_i} L()$ take the form:

$$L(\cdot) = \sum_{i=1}^n \left(\frac{1}{n} f(\mathbf{b}_i) + h_i(\mathbf{y}_i) + \boldsymbol{\mu}_i^\top (\mathbf{b}_i - \mathbf{b}) + \boldsymbol{\lambda}_i^\top (\mathbf{b}_i - \mathbf{y}_i) \right)$$

$$q() = \sum_{i=1}^n \left(\min_{\mathbf{b}} \left(\frac{1}{n} f(\mathbf{b}) + \mathbf{b}^\top (\boldsymbol{\lambda}_i + \boldsymbol{\mu}_i) \right) - h_i^*(\boldsymbol{\lambda}_i) \right)$$

Note that whenever $\sum_{i=1}^n \boldsymbol{\mu}_i \neq 0$ the dual function attains the value $q() = -\infty$. Since we seek to maximize the dual function we need to optimize $\boldsymbol{\mu}_i$ in its domain, i.e. $\sum_{i=1}^n \boldsymbol{\mu}_i = 0$. □

The function $\sum_{i=1}^n \frac{1}{n} f(\mathbf{b}_i)$ is strictly convex therefore its conjugate dual is differentiable (see (Rockafellar, 1970)). In this case a block dual ascent optimization scheme converges to the global maxima (Tseng, 1993). Our algorithm alternates through optimizing $\boldsymbol{\lambda}_i$ (in parallel) while fixing $\boldsymbol{\mu}_i$ followed by optimizing $\boldsymbol{\mu}_i$ (by a closed form solution) while fixing $\boldsymbol{\lambda}_i$. We formulate our dual algorithmic building block with respect to $\boldsymbol{\lambda}_i$ using Theorem 1: Set $g(\mathbf{b}) \leftarrow \frac{1}{n} f(\mathbf{b}) + \mathbf{b}^\top \boldsymbol{\mu}_i$ and $h(\mathbf{b}) \leftarrow h_i(\mathbf{b})$, and solve the primal program:

$$\mathbf{b}^* = \operatorname*{argmin}_{\mathbf{b} \in domain(h_i)} \left\{ \frac{1}{n} f(\mathbf{b}) + \mathbf{b}^\top \boldsymbol{\mu}_i + h_i(\mathbf{b}) \right\}$$

From Lagrange optimality condition in Theorem 1 we recover $\boldsymbol{\lambda}_i^*$

$$\boldsymbol{\lambda}_i^* = -\boldsymbol{\mu}_i - \frac{1}{n} \nabla f(\mathbf{b}^*) \quad (9)$$

We turn to find the closed-form solution for optimizing $\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_n$ while fixing $\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_n$ using Theorem 1: Set $g(\mathbf{b}_1, ..., \mathbf{b}_n) \leftarrow \frac{1}{n} \sum_{i=1}^n (f(\mathbf{b}_i) + \mathbf{b}_i^\top \boldsymbol{\lambda}_i)$ and set $h(\mathbf{b}_1, ..., \mathbf{b}_n)$ to be the indicator function that attains the value zero if $\mathbf{b}_1 = \cdots = \mathbf{b}_n$ and infinity otherwise. The conjugate function of $h(\mathbf{b}_1, ..., \mathbf{b}_n)$ is the indicator function $h^*(\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_n)$ whose value is zero if $\sum_{i=1}^n \boldsymbol{\mu}_i = 0$ and ∞ otherwise. The primal program:

$$\operatorname*{argmin}_{\mathbf{b}_1, ..., \mathbf{b}_n \in domain(h)} \left\{ \sum_{i=1}^n \left(\frac{1}{n} f(\mathbf{b}_i) + \mathbf{b}_i^\top \boldsymbol{\lambda}_i \right) \right\}$$

can be further simplified by taking into account the domain of $h(\mathbf{b}_1, ..., \mathbf{b}_n)$, i.e. restricting all the $\mathbf{b}_i$ to equal some vector $\mathbf{b} \in \mathbb{R}^n$:

$$\operatorname*{argmin}_{\mathbf{b} \in \mathbb{R}^n} \left\{ f(\mathbf{b}) + \mathbf{b}^\top \sum_{i=1}^n \boldsymbol{\lambda}_i \right\}$$

Since $f(\mathbf{b})$ is real-valued function and the optimization is unconstrained the optimal vector $\mathbf{b}^*$ satisfies $\nabla f(\mathbf{b}^*) = -\sum_{i=1}^n \boldsymbol{\lambda}_i$.
Theorem 1 asserts the Lagrange multipliers $\boldsymbol{\nu}^* = \boldsymbol{\mu}_1^*, ..., \boldsymbol{\mu}_n^*$ equals the gradient of $g(\mathbf{b}_1^*, ..., \mathbf{b}_n^*)$, or equivalently $\boldsymbol{\mu}_i^* = -\frac{1}{n} \nabla f(\mathbf{b}_i^*) - \boldsymbol{\lambda}_i$. In the preceding paragraph we argued that $\nabla f(\mathbf{b}^*) = -\sum_{i=1}^n \boldsymbol{\lambda}_i$ so we derive the update rule for $\boldsymbol{\mu}_i^*$

$$\boldsymbol{\mu}_i^* = -\boldsymbol{\lambda}_i + \frac{1}{n} \sum_{i=1}^n \boldsymbol{\lambda}_i \quad (10)$$

Taken together, one obtains Algorithm 2 described in Section 3.

Learning When to Take Advice: A Statistical Test for Achieving A Correlated Equilibrium

Greg Hines
Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
ggdhines@cs.uwaterloo.ca

Kate Larson
Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
klarson@cs.uwaterloo.ca

Abstract

We study a multiagent learning problem where agents can either learn via repeated interactions, or can follow the advice of a mediator who suggests possible actions to take. We present an algorithm that each agent can use so that, with high probability, they can verify whether or not the mediator's advice is useful. In particular, if the mediator's advice is useful then agents will reach a correlated equilibrium, but if the mediator's advice is not useful, then agents are not harmed by using our test, and can fall back to their original learning algorithm. We then generalize our algorithm and show that in the limit it always correctly verifies the mediator's advice.

1 Introduction

In settings where agents repeatedly interact with each other (for example, through a repeated game), there are great opportunities for learning since agents are able to adapt their strategies given the history of play. This problem has garnished a lot of attention from several research communities, including the AI community and the game theory community. While many criteria have been proposed for measuring the success of learning approaches, one commonly used measure is whether the agents learn how to best-respond to the strategies being played by the others. That is, does the learning process converge to an equilibrium.

In this paper we study the problem of agents interacting with each other in a repeated game setting, but we introduce a third party *mediator* or *advisor* who makes strategy suggestions to the agents. Ideally, by following the suggestions of the mediator, agents will be able to learn how to play against each other, possibly even reaching mutually beneficial outcomes which would not have been possible without the mediation. That is, our goal is for the agents to learn and adapt so that they find a correlated equilibrium [1].

However, a mediator is only useful if it can make good suggestions. Even if a mediator tries to make good suggestions it may be prevented by coding errors, memory limitations, *etc*. For an agent to accept a mediator's suggestions, there must be some way for the agent to verify that the suggestions are reasonable. A mediator might not be willing to share its code with the agents, or be aware of its own limitations. Therefore, for a truly robust system, the agents themselves must have a way of checking the mediator's suggestions.

Thus, this paper introduces a statistical test based on hypothesis testing that, with high probability, can verify the mediator's suggestions. While hypothesis testing has been proposed in the multiagent learning literature as a tool that agents might use to learn how to play Nash equilibria [5], to the best of our knowledge it has never been applied for validating a mediator's advice. Based on our test, we propose an algorithm that allows agents to converge to the mediator's suggestion if it is a correlated equilibrium and otherwise, in the limit, be no worse off for having used our algorithm. We then generalize this algorithm to a more theoretical setting where we show that with probability one, in the limit, our test will always be able to correctly verify the mediator's suggestions. This provides a method for achieving convergence to a specific correlated equilibrium.

2 Background

In this section we introduce the key concepts and assumptions used in this paper.

A n-agent *stage game* is a tuple $G = \langle N, A = A_1 \times \ldots \times A_n, u_1, \ldots, u_n \rangle$, where $N = \{1, \ldots, n\}$ is the set of agents, A_i is the set of possible actions for agent i and A is the set of possible joint actions, and $u_i : A \rightarrow \mathbb{R}$ is the utility function for agent i. Without loss of generality, we assume that all utilities are greater than or equal to 0. A specific action for agent i is $a_i \in A_i$, and a joint action is $a = (a_1, \ldots, a_n)$. We assume that A is public knowledge but the agents' utility functions are private.

Each agent chooses its actions according to some *strategy*. A strategy for agent i, σ_i, is a probability distribution over A_i, stating with what probability the agent will play each

possible action. The set of all possible strategies for agent i is Σ_i. The vector $\sigma = (\sigma_1, \ldots, \sigma_n)$ is a strategy profile which specifies a strategy for each agent and Σ is the set of all possible strategy profiles. We use σ_{-i} to denote $(\sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_n)$.

Given a strategy profile σ, we define the *expected utility* for agent i as

$$u_i(\sigma) = \sum_{a=(a_1,\ldots,a_n)\in A} u_i(a)\Pi_{j=1}^{n}\sigma_j(a_j). \quad (1)$$

Each agent's utility is dependent not just on its own actions, but also on the actions taken by all other agents. We assume agents are *rational*, *i.e.*, given σ_{-i}, agent i will choose a strategy which maximizes its expected utility.

In our model we introduce a third-party mediator, $\mathcal{M}$. The mediator knows the utility functions for all agents, but is not affected by the game's outcome. Instead $\mathcal{M}$ makes suggestions to each agent as to what action it should take, where these suggestions are instantiations of a correlated strategy.

Definition 1. *A* correlated strategy, σ_A, *is a probability distribution over A. We let $s \in A$ denote an instantiation of σ_A. The* conditional correlated strategy $\sigma_{A_{-i}}(s_{-i}|s_i)$ *is the conditional probability of the joint signal (s_i, s_{-i}) given the signal s_i, and $\sigma_{A_{-i}}(s_i)$ is the set of all conditional probabilities given s_i.*

Note that σ_i is a probability distribution over A_i while σ_A is a probability distribution over A.

We assume that $\mathcal{M}$'s correlated strategy is public knowledge, but the actual instantiation, s, is not. In particular we assume that $\mathcal{M}$ sends each agent i a private signal, s_i, based on s.

The agents are under no obligation to follow the mediator's signals. It is up to the mediator to pick a correlated strategy that a rational agent would be willing to follow. Note that our type of a mediator is different than Monderer and Tennenholtz's, where agents must agree to follow the mediator's suggested actions before knowing what they are [11].

Definition 2. *A correlated strategy $\sigma_A^* = \{\sigma_A(a)|a \in A\}$ is a correlated equilibrium if for every agent i and every $s_i \in A_i$,*

$$\sum_{s_{-i}\in A_{-i}} \sigma^*_{A_{-i}}(s_{-i}|s_i)u_i(s_i, s_{-i}) \quad (2)$$
$$\geq \sum_{s_{-i}\in A_{-i}} \sigma^*_{A_{-i}}(s_{-i}|s_i)u_i(a_i', s_{-i}),$$

for all $a_i' \in A_i$ [1]. The set of all correlated equilibria in G is $C(G)$.

If all of agent i's opponents are following a correlated equilibrium σ_A^*, it is rational for agent i to also follow σ_A^*.

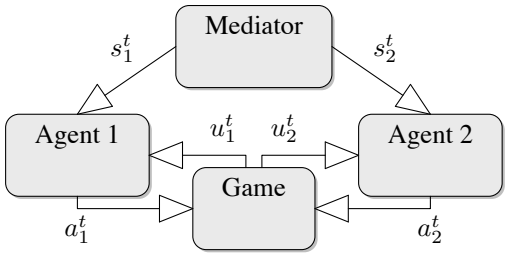


Figure 1: A graphical representation of our setting with 2 agents at time t.

In this paper, we are interested in a setting where agents have the ability to learn and adapt to the actions taken by others. Thus, we study repeated games. A repeated game $G^r = (G^1, G^2, \ldots)$ is an infinite sequence of the stage game G played repeatedly. Agent i's action at time t is a_i^t and the joint action at time t is a^t. The history of joint actions, $hist(t) = \{a^1, \ldots, a^{t-1}\}$, is a record of the joint action taken at each iteration until time t. The empirical, or observed, percentage of play of joint actions, $\sigma_A^{hist(t)}$, is the percentage of time each joint action has been played as of time t. Agents may learn from previous iterations of the game to try and improve their strategy. Specifically, we assume that agent i has a learning algorithm $L_i : hist(t) \rightarrow \Sigma_i$, that helps agent i select a strategy for time t.

Let σ_A^t be the actual correlated strategy at time t, *i.e.* the one agents are actually using and not necessarily the one based on $\mathcal{M}$'s suggestions. We say that σ_A^t converges to a correlated equilibrium if for some $\sigma_A^* \in C(G)$, $\lim_{t\rightarrow\infty} \sigma_A^t = \sigma_A^*$. Thus, our algorithm is differentiated from algorithms that achieve convergence to the set of correlated equilibrium, for example [4, 8].

3 Setup

The setting for our paper is a repeated game G^r with a mediator, $\mathcal{M}$. As illustrated for the two agent case in Figure 1, time t will begin with the mediator giving each agent a suggested action, s_i^t. Agents will then simultaneously choose their action, a_i^t, which may or may not be s_i^t. If agent i chooses not to follow $\mathcal{M}$'s signal, it can instead use a learning algorithm, L_i, which we assume is independent of $\mathcal{M}$'s signals, to select an action. Based on the actual joint action, each agent will then receive some utility and the process repeats. The mediator's signal to each agent is private information, known only to that agent and the mediator, as is the agent's utility function. However, the action set for each agent is public knowledge, as is the action taken by each agent during a turn.

The mediator's signals are based on a selected correlated strategy, $\sigma_A^{\mathcal{M}}$, which is constant throughout the repeated game. Although ideally the mediator will suggest a correlated strategy that is also a correlated equilibrium, each

agent still needs to verify that the mediator has actually done so.

Our aim is to design an algorithm that achieves the following goals.

First goal: If $\sigma_A^{\mathcal{M}}$ is a correlated equilibrium then σ_A^t, the actual correlated strategy which is not necessarily $\sigma_A^{\mathcal{M}}$, will converge to $\sigma_A^{\mathcal{M}}$.

Second goal: If $\sigma_A^{\mathcal{M}}$ is not a correlated equilibrium, agents should be no worse off, in the limit, for having used our algorithm.

In Section 4, we present an algorithm, Λ, that achieves these goals with high probability. In Section 5, we generalize Λ so that, with probability one, in the limit, it will achieve both goals. Since each agent will be using Λ independently, we refer to Λ_i as the instance of the algorithm being run by agent i and Λ as the joint algorithm.

The algorithm is based on the concept of giving $\mathcal{M}$ the benefit of the doubt; until there is reason to believe otherwise, agents assume that $\sigma_A^{\mathcal{M}}$ is a correlated equilibrium and follow $\mathcal{M}$'s signals. Specifically, agents will assume that the following conditions hold.

Condition 1: The correlated strategy $\sigma_A^{\mathcal{M}}$ is a correlated equilibrium.

Condition 2: All other agents are following the signals based on $\sigma_A^{\mathcal{M}}$.

Agents test whether these conditions hold during an initial period of play called a *sampling test* which has a fixed length of l_T. If, at the beginning of the sampling test, agent i decides that one of the conditions does not hold, it will not follow $\mathcal{M}$'s signals and instead will use an individual "fall-back" strategy, γ_i, chosen uniformly at random. At the end of the sampling test, all agents who still believe that both conditions hold will continue to follow $\mathcal{M}$'s signals. All other agents will start using their original learning algorithm. The algorithm Λ_i is correct if and only if, at the end of the sampling test, it correctly determines whether both conditions hold. The joint algorithm, Λ, is correct if and only if Λ_i is correct for all i.

4 The Initial Algorithm

In this section, we describe how our initial algorithm works. As a first step in Λ_i, agent i will check to see if Equation 2 holds for all $s_i \in A_i$. If Equation 2 does not hold, agent i will know that Condition 1 cannot be true. In this case, agent i will use a "fall-back" strategy, $\gamma_i \in \Sigma_i$, picked uniformly at random, for the rest of the sampling test. If Equation 2 does hold, agent i must check to see if Condition 2 is true and will continue to follow $\mathcal{M}$'s signals throughout the sampling test.

Since the utilities for each agent, as well as the signals they receive each turn, are private, there may be no way to prove or disprove Condition 2 with absolute certainty at any finite point during the game. The best Λ_i can do is reach a probabilistic conclusion. Since joint actions are public knowledge, Λ_i can compare the empirical percentages of play for the duration of the sampling test against the percentages predicted by $\sigma_A^{\mathcal{M}}$. If the difference between these two values is statistically significant, there is a high probability that at least one agent has stopped following the mediator's signals.

To test if there is a difference, agent i assumes there is some fixed but unknown correlated strategy $\tilde{\sigma}_A$ that all agents were actually using for the sampling test, where $\tilde{\sigma}_A$ may or may not be $\sigma_A^{\mathcal{M}}$. We are now able to use hypothesis testing, where our null hypothesis is that $\sigma_A^{\mathcal{M}}$ is equal to $\tilde{\sigma}_A$, *i.e.*,

$$H_0 : \sigma_A^{\mathcal{M}} = \tilde{\sigma}_A, \tag{3}$$

and our alternative hypothesis is that $\sigma_A^{\mathcal{M}}$ is not equal to $\tilde{\sigma}_A$, *i.e.*,

$$H_1 : \sigma_A^{\mathcal{M}} \neq \tilde{\sigma}_A. \tag{4}$$

The test statistic used is Pearson's χ^2 test,

$$\mathcal{T} = \sum_{a \in A'} \frac{(X(a) - E(a))^2}{E(a)}, \tag{5}$$

where A' is any subset of A such that $|A'| = |A| - 1$, $X(a) = l_T \sigma_A^{hist(l_T)}(a)$ is the actual frequency of play of $a \in A'$ during the sampling test, $E(a) = l_T \sigma_A^{\mathcal{M}}(a)$ is the expected frequency of play according to $\sigma_A^{\mathcal{M}}$, and where l_T is the length of the sampling period [12]. Note that $\sigma_A^{hist(l_T)}$ is based on a sampling from $\tilde{\sigma}_A$ of size l_T. For now we assume that $\sigma_A^{\mathcal{M}}(a) > 0$ for all $a \in A$. We relax this assumption later. The Pearson's χ^2 test has (in the limit) a probability distribution function of

$$\chi^2_{df} + \chi^2_{NCP,1}, \tag{6}$$

where the first distribution has $df = |A| - 2$ degrees of freedom, and the second distribution has 1 degree of freedom and a non-centrality parameter of NCP [9].

If H_0 is true, $NCP = 0$. Assuming that H_0 is true, we choose a significance level for rejection of the null hypothesis of $\alpha < 1$ and a corresponding critical value of $c(\alpha)$, *i.e.,* we reject the null hypothesis when $\mathcal{T} \geq c(\alpha)$. In this case, the probability of incorrectly rejecting H_0 (known as a Type 1 error) is $p_1 = \alpha$. If H_1 is actually true, we err when $\mathcal{T} < c(\alpha)$ and we do not reject H_0 (a Type 2 error). When H_1 is true, $NCP > 0$. Since the non-centrality parameter determines how much the probability distribution in Equation 6 gets adjusted, determining NCP helps determine the probability of a Type 2 error.

The equation for NCP is $NCP = t * \delta$, where δ, the sensitivity parameter, is a measure of the difference between $\sigma_A^{\mathcal{M}}$ and $\tilde{\sigma}_A$ given by

$$\delta(\sigma_A^{\mathcal{M}}, \tilde{\sigma}_A) = \sum_{a \in A} \frac{(\tilde{\sigma}_A(a) - \sigma_A^{\mathcal{M}}(a))^2}{\sigma_A^{\mathcal{M}}(a)}. \tag{7}$$

For a given value of δ, say $\hat{\delta}$, if

$$\delta(\sigma_A^{\mathcal{M}}, \tilde{\sigma}_A) \geq \hat{\delta}, \tag{8}$$

then the probability of a Type 2 error is bounded by some value $\beta(\hat{\delta}) < 1$, whose value is normally found via numerical computation [9]. Since β is also a function of l_T and α, we refer to it as $\beta(l_T, \alpha, \delta)$.

Since agents do not know whether their opponents are following the mediator's suggestions, agents do not know the exact value for $\tilde{\sigma}_A$, and therefore, it is impossible to choose an appropriate value for $\hat{\delta}$ so that Equation 8 is guaranteed to hold. Instead, agents can consider a different question: what is the worst case situation under which Equation 8 does not hold? To answer this question, consider the set of all agents for whom Equation 2 does not hold, $N_B \subseteq N$. Let $(\sigma^{\mathcal{M}}_{A_{-N_B}}, \gamma_{N_B})$ be the actual correlated strategy for the duration of the sampling test, *i.e.*, a combination of those agents who will follow $\mathcal{M}$'s signals and those who will rely on their fall-back strategy. Let Σ_{N_B} be the set of all possible joint strategies for agents in N_B, and

$$\Sigma_{N_B}(\sigma_A^{\mathcal{M}}, \delta) = \{\gamma_{N_B} \in \Sigma_{N_B} | \delta(\sigma_A^{\mathcal{M}}, (\sigma^{\mathcal{M}}_{A_{-N_B}}, \gamma_{N_B})) < \delta\} \tag{9}$$

be the set of all possible joint strategies for agents in N_B which would result in Equation 8 not holding. Let $\mu(\Sigma_{N_B})$ and $\mu(\Sigma_{N_B}(\sigma_A^{\mathcal{M}}, \delta))$ be the Lebesgue measures of Σ_{N_B} and $\Sigma_{N_B}(\sigma_A^{\mathcal{M}}, \delta)$, respectively. Then, since γ_i is chosen uniformly at random, the probability of σ_{N_B} being in $\Sigma_{N_B}(\sigma_A^{\mathcal{M}}, \delta)$ is

$$\psi(\Sigma_{N_B}) = \frac{\mu(\Sigma_{N_B}(\sigma_A^{\mathcal{M}}, \delta))}{\mu(\Sigma_{N_B})}. \tag{10}$$

Since agents do not know N_B, they consider the worst case scenario,

$$\psi = \max_{N' \subseteq N} \psi(\Sigma_{N'}). \tag{11}$$

If we assume that whenever Equation 8 does not hold and $\tilde{\sigma}_A \neq \sigma_A^{\mathcal{M}}$, a Type 2 error is always made, then the probability of a Type 2 error is at most

$$p_2 \leq (1 - \psi) \cdot \beta(\hat{\delta}) + \psi. \tag{12}$$

That is, Equation 8 holds with at least a probability of ψ and when it does, the probability of a Type 2 error is at most $\beta(\hat{\delta})$ and with a probability of at most ψ, Equation 8 does not hold.

If we do not assume that $\sigma_A^{\mathcal{M}}(a) > 0$ for all $a \in A$, then Equations 5 and 7 may contain division by zero. To deal with this, we ignore all $a \in A$ such that $\sigma^{\mathcal{M}}(a) = 0$. If $\zeta = \{a \in A | \sigma^{\mathcal{M}}(a) = 0\}$, then the summations in Equations 5 and 7 need to exclude all $a \in \zeta$, and *df* in Equation 6 now equals $|A| - 2 - |\zeta|$. If the null hypothesis is correct then $\sigma_A^{\mathcal{M}}(a) = 0$ implies that $\sigma_A^{hist(l_T)}(a) = 0$ for all $a \in \zeta$. Alternatively, if there exists $a' \in A$ such that $\sigma_A^{hist(l_T)}(a') > 0$ while $\sigma_A^{\mathcal{M}}(a') = 0$, the alternative hypothesis must be correct. Hence, both of these cases do not present problems.

The only other case is if for all $a \in A$ such that $\sigma_A^{\mathcal{M}}(a) = 0$, $\sigma_A^{hist(l_T)}(a) = 0$ but, unknown to the agents, the alternative hypothesis is correct. In this case, a Type 2 error may occur. To find the probability of this case happening, we first determine the probability of $a^t \in \zeta$. Since any agent who rejects $\mathcal{M}$'s suggested strategy chooses its new strategy uniformly at random, the probability, $\mathcal{P}$, that $a^t \in \zeta$ for $t \leq l_T$ is

$$\mathcal{P} \geq \sum_{a \in \zeta} \min_{N' \subseteq N} \sigma_{A_{-N'}}(a_{-N'}) \frac{1}{|A_{N'}|}, \tag{13}$$

where $\min_{N' \subseteq N}$ is considered since agents do not know N_B. Therefore, the probability that $a^t \notin \zeta$ for all $t \leq l_T$ is at most $(1 - \mathcal{P})^{l_T}$ and the overall probability of a Type 2 error is at most

$$p_2 \leq (1 - \mathcal{P})^{l_T} \left[(1 - \psi) \cdot \beta + \psi\right]. \tag{14}$$

To accommodate the worst case, we assume equality holds in Equation 14. Note that p_1 has not changed. For simplicity, we assume that $p_1 = p_2 = p$, and refer to p as the overall probability of error.

It is possible to rearrange $\beta(l_T, \alpha, \delta)$ to express l_T as a function of α, β and δ, *i.e* $l_T(\alpha, \beta, \delta)$. As a result, l_T is the sample size needed to perform the test with at most a probability of error (of either Type 1 or Type 2) of p.

If all agents are to use the same value for l_T, they must also have the same value for β. This in turn requires them to have the same value for ψ. To achieve this, in Equations 11 and 13, agent i will consider all possible N', including those containing agent i.

4.1 Examples

In this section we provide two examples to illustrate how our test would work.

Example 1: Consider the game in Figure 2.

		Agent 2	
		$a_{2,1}$	$a_{2,2}$
Agent 1	$a_{1,1}$	0,1	2,5
	$a_{1,2}$	5,2	1,0

Figure 2: A simple game

Let $A = \{(a_{1,1}, a_{2,1}), (a_{1,1}, a_{2,2}), (a_{2,1}, a_{2,1}), (a_{1,2}, a_{2,2})\}$. Suppose that $\mathcal{M}$ announces a correlated strategy, $\sigma_A^{\mathcal{M}} = \{1/18, 5/18, 2/18, 10/18\}$. Note that $\sigma_A^{\mathcal{M}}$ is a correlated equilibrium.

Suppose the agents choose $p = 0.1$ and $\delta = 0.01$. Agents must now determine the critical value for rejection, $c(\alpha)$,

and the length of the sampling test, l_T. Since $p_1 = \alpha$, $\alpha = 0.1$. For 3 degrees of freedom, $c(\alpha) = 6.25$. Since $\sigma_A^{\mathcal{M}}(a) > 0$ for all a, we can calculate β by Equation 12. We calculate Equation 11 by numerical computation to find $\psi \approx 0.09429$. Therefore, $\beta = 0.0063$. In practice, $l_T(\alpha, \beta, \delta)$ would now be solved by some method of numerical computation [9]. For simplicity, we used the tables in Cohen to obtain a value of $l_T = 2100$ [2].

Suppose that after 2100 iterations, we have obtained an empirical frequency of play $\theta_A^{hist(2101)} = \{96, 601, 224, 1179\}$. Using Equation 5, we obtain a test statistic value of 4.678. Since this is lower than the critical value, both agents do not reject the null hypothesis and continue to use $\mathcal{M}$'s signals.

Example 2: Consider a different example based on the same game where $\mathcal{M}$ announces a correlated strategy of $\sigma_A^{\mathcal{M}} = \{2/18, 10/18, 1/18, 5/18\}$. In this case, $\sigma_A^{\mathcal{M}}$ is not a correlated equilibrium. Specifically, while Equation 2 is satisfied for Agent 1, it is not satisfied for Agent 2. Hence, Agent 2 will use a random fall-back strategy. Suppose $\gamma_2 = (3/4, 1/4)$.

For this example, the length of the test has not changed. Suppose we find an empirical frequency of $\theta_A^{hist(2101)} = \{1050, 350, 525, 175\}$ after 2100 turns. Since Agent 2 already knows that $\sigma_A^{\mathcal{M}}$ is not a correlated equilibrium, it will not perform the test. Agent 1 will obtain a test statistic value of 5953.3. This is well above the critical value and so Agent 1 will reject the null hypothesis, *i.e.*, it will stop following the signals of the mediator.

Note that, as we have stated our algorithm, Agent 1 will only know that there is a probability of at most 0.1 of incorrectly rejecting the null hypothesis. We have not accounted for the fact that the test statistic value is much higher than the critical value. An additional test that could be run after the null hypothesis is rejected is the calculation of the *p-value*. The p-value is the smallest α value that would still allow us to reject the hypothesis [12]. In the case of the above example, the p-value would be very small, and Agent 1 could be very certain that $\sigma_A^{\mathcal{M}}$ is not a correlated equilibrium.

5 Repeated Testing

The limitation of our basic test is that there is always some positive probability of error. This is due to the need to pick values for $1-p$ and δ that are both greater than 0. Since we can pick any such values for $1-p$ and δ, this is not much of a practical limitation, however we may wish to achieve a stronger theoretical result. Our goal is to have agents converge to playing $\sigma_A^{\mathcal{M}}$ if it is a correlated equilibrium. If $\sigma_A^{\mathcal{M}}$ is not a correlated equilibrium, then the agents' utility should be no worse off for having used our algorithm. This leads to the idea of *repeated testing*, where throughout the repeated game, agents will use multiple iterations of Λ_i.

The set of repeated sampling tests is $R = \{R_1, R_2, \ldots\}$,

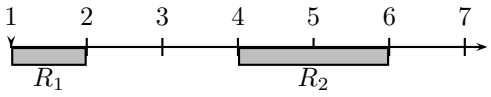


Figure 3: An example of repeated testing.

where $R_j = \{b_{R_j}, l_{R_j}\}$, b_{R_j} is the first time period in R_j, and l_{R_j} is the length of R_j. The instance of Λ_i during test R_j is denoted by $\Lambda_i^{R_j}$. The repeated tests are not contiguous. A simple example is shown in Figure 3, where the timeline represents a repeated game up to 7 iterations. The grey areas represent sampling test iterations. For example, $R_2 = \{b_{R_2}, l_{R_2}\} = \{4, 2\}$, meaning that the second test iteration begins at time period 4 and lasts for 2 iterations of the repeated game.

The parameters, δ and p, can be set to depend on the test iteration, *i.e.* $\delta(R_j)$ and $p(R_j)$. Each test period must be identical for each agent, *i.e.* R_j must be the same for all agents. This means that $\delta(R_j)$ and $p(R_j)$ must be the same for all agents. The parameters are chosen such that

$$\lim_{j \to \infty} \delta(R_j) = 0, \tag{15}$$

$$\sum_{j=1}^{\infty} p(R_j) < \infty. \tag{16}$$

For example, we can let $\delta(R_j) = 1/j$ and $p(R_j) = 1/2^j$. Finally, we assume that each agent's fall-back strategy is fixed. That is $\gamma_i^{R_j} = \gamma_i^{R_{j'}}$, for all j, j'.

Our first result is that an agent will not draw the wrong conclusion about the mediator too often.

Theorem 1. *In the limit, with probability one, there will only be a finite number of tests where Λ^{R_j} is incorrect.*

Proof. Let $\sigma_A^{\mathcal{M}}$ be the correlated strategy suggested by $\mathcal{M}$. Consider the following two cases:

$\sigma_A^{\mathcal{M}}$ is a correlated equilibrium: For test R_j, the probability of $\Lambda_i^{R_j}$ making a Type 1 error, $p_1(R_j)$, is equal to $p(R_j)$. By the Borel-Cantelli lemma, with probability one, there will only be a finite number of times $\Lambda_i^{R_j}$ is incorrect, *i.e.* makes a Type 1 error. [1] This reasoning can be applied to all agents, and therefore with probability one there will only be a finite number of times Λ^{R_j} is incorrect.

$\sigma_A^{\mathcal{M}}$ is not a correlated equilibrium: If $\sigma_A^{\mathcal{M}}$ is not a correlated equilibrium, then some subset of agents, $N' \subseteq N$, will use their fall-back strategies instead of following the mediator's signals. The resulting correlated strategy for every test iteration will be $(\sigma_{A_{-N'}}^{\mathcal{M}}, \gamma_{N'})$.

Since $\gamma_{N'}$ is fixed, by Equation 15, there exists a finite j^*

[1] **Borel-Cantelli Lemma:** Let $\{E^t\}_0^\infty$ be a sequence of independent events and $P(E^t)$ be the probability of the event E^t occurring. If $\sum_{t=0}^{\infty} P(E^t) < \infty$, then with probability one, only a finite number of the events will occur.

such that for all $j \geq j^*$,

$$\delta(\sigma_A^{\mathcal{M}}, (\sigma_{A_{-N'}}^{\mathcal{M}}, \gamma_{N'})) \geq \delta(R_j). \tag{17}$$

Let $\psi(R_j)$ be the value of ψ, according to Equation 11, during the sampling test R_j. Starting at R_{j^*}, we know that, with probability one, Equation 8 holds and therefore, since $\psi(R_j)$ is the probability of Equation 8 not holding, $\psi(R_j) = 0$, for all $j \geq j^*$. Therefore, the probability of a Type 2 error starting at R_{j^*} is

$$p_2 = \sum_{j=j^*}^{\infty} (1-\mathcal{P})^{l_T} \beta. \tag{18}$$

Note that $\mathcal{P}$, l_T and β are all functions R_j, however we omit the notation (R_j) for clarity. Since β is less than 1,

$$p_2 \leq \sum_{j=j^*}^{\infty} (1-\mathcal{P})^{l_T} \left[(1-\psi)\cdot\beta + \psi\right] \tag{19}$$

$$= \sum_{j=j^*}^{\infty} p(R_j), \tag{20}$$

where ψ, as calculated by Equation 11, is also a function of R_j. Therefore, by Equation 16 and the Borel-Cantelli lemma, with probability one, there will only be a finite number of times $\Lambda_i^{R_j}$ is incorrect, *i.e.* makes a Type 2 error. Again, this reasoning can be generalized to all agents and therefore, there will only be a finite number of times Λ^{R_j} is incorrect. □

We now examine the behaviour of agents between sampling tests. The periods between test iterations are called *free periods*. The set of free periods is $F = \{F_1, \ldots\}$ where $F_j = \{b_{F_j}, l_{F_j}\}$. Thus $G^r = \{R_1, F_1, R_2, F_2, \ldots\}$. For example, in Figure 3, the first free period, F_1, would be $\{b_{F_1}, l_{F_1}\} = \{2, 2\}$. If $\Lambda_i^{R_j}$ did not reject the null hypothesis, agent i continues to follow $\mathcal{M}$'s signals for all of F_j. If $\Lambda_i^{R_j}$ did reject the null hypothesis, agent i relies on its learning algorithm L_i for F_j. We assume that L_i is *flexible* at the beginning of each free period [3].

Definition 3. *The learning algorithm L_i is flexible if at the beginning of every free period F_j,*

$$L_i(hist(b_{F_j})) = L_i(hist(1)). \tag{21}$$

Therefore, during each free period, L_i does not base its actions on what has happened before time b_{F_j}.

For example, L_i may be a trigger strategy, but that trigger may not be based on anything that has happened in a previous sampling test or free period.

We require that

$$\lim_{j\to\infty} \frac{\sum_{j'=1}^{j} l_{R_j}}{\sum_{j'=1}^{j} l_{F_j}} = 0, \tag{22}$$

for example $l_{F_j} = l_{R_j}^2$. This means that, in the limit, the length of the sampling periods is negligible compared to the length of the free periods. We also require that

$$\lim_{j\to\infty} \frac{l_{R_j}}{j} = \infty. \tag{23}$$

This means that the length of the sampling tests grows at faster than a linear rate. The specific values for l_{R_j} and l_{F_j} would have to be agreed upon by all agents.

Definition 4. *Let $\theta_A^{exp(t_1,t_2)}$ be the expected frequency of play from time t_1 to t_2, i.e., the expected number of times each joint action $a \in A$ gets played between times t_1 and t_2 inclusive. If t_1 is not given, we assume $t_1 = 1$. Similarly, let $\theta_A^{exp(F_j,\ldots,F_{j'})}$ be the expected frequency of play during the free periods F_j through $F_{j'}$, inclusive.*

Since the frequency of play depends on the algorithms the agents are using, let $\theta_A^{exp(t)}(L)$ be the expected frequency of play from time 1 to t assuming that agents use the joint learning algorithm L for the whole period.

For simplicity in all of the following proofs, we assume that t always corresponds to the beginning of a sampling period. Let $j(t)$ be the index of the last free period before t.

The first step is to show that if $\mathcal{M}$ suggests a correlated equilibrium, agents will converge to it.

Theorem 2. *If the correlated strategy suggested by $\mathcal{M}$, $\sigma_A^{\mathcal{M}}$, is a correlated equilibrium, then with probability one,*

$$\lim_{t\to\infty} \sigma_A^t = \sigma_A^{\mathcal{M}}. \tag{24}$$

Proof. If $\sigma_A^{\mathcal{M}}$ is a correlated equilibrium then by Theorem 1, with probability one, after some finite point Λ will always correctly determine that $\sigma_A^{\mathcal{M}}$ is a correlated equilibrium. As a result, with probability one, after some finite point, all agents will choose to follow the mediator's signals during the free periods. □

Our next result is a technical lemma which shows that in the limit, agents are not harmed by taking time out to do the sampling tests.

Lemma 1. *In the limit, there is no difference between the average utility from agents using L for the whole repeated game and just for the free periods, i.e.,*

$$\lim_{t\to\infty} \left[u_i\left(\frac{\theta_A^{exp(t)}(L)}{t}\right) - u_i\left(\frac{\theta_A^{exp(F_1,\ldots,F_{j(t)})}(L)}{t}\right)\right] = 0. \tag{25}$$

Furthermore, this is true even when excluding the first $j^* - 1$ free periods, for some $j^* > 1$, i.e.,

$$\lim_{t\to\infty} \left[u_i\left(\frac{\theta_A^{exp(t)}(L)}{t}\right) - u_i\left(\frac{\theta_A^{exp(F_{j^*},\ldots,F_{j(t)})}(L)}{t}\right)\right] = 0. \tag{26}$$

The proof is given in the Appendix.

Finally, we need to show that if $\sigma_A^{\mathcal{M}}$ is not a correlated equilibrium, agents are no worse off, on average, for having used Λ.

Theorem 3. *If the correlated strategy suggested by $\mathcal{M}$, $\sigma_A^{\mathcal{M}}$, is not a correlated equilibrium, then with probability one,*

$$\lim_{t\to\infty}\left[u_i\left(\frac{\theta_A^{exp(t)}(\Lambda)}{t}\right) - u_i\left(\frac{\theta_A^{exp(t)}(L)}{t}\right)\right] \geq 0. \quad (27)$$

Therefore, in the limit, agent i will be no worse off for using Λ instead of L_i.

Proof. If $\sigma_A^{\mathcal{M}}$ is not a correlated equilibrium, by Theorem 1, with probability one, starting at some sampling test, say R_{j^*}, Λ will always correctly determine that $\sigma_A^{\mathcal{M}}$ is not a correlated equilibrium.

Consider θ_A with respect to some arbitary $a \in A$, denoted by θ_a. We start by breaking the game down into the sequence of sampling tests and free periods. That is, $\theta_a^{exp(t)}(\Lambda) = \theta_a^{exp(R_1,F_1,\ldots,F(t))}(\Lambda)$. For $t \geq t(j^*)$, the utility can be split up into the utility for the sampling tests and free periods before R_{j^*} and for those starting at R_{j^*} *i.e.*,

$$\lim_{t\to\infty}\left[u_i\left(\frac{\theta_a^{exp(R_1,F_1,\ldots,R_{j^*-1},F_{j^*-1})}(\Lambda)}{t}\right) + u_i\left(\frac{\theta_a^{exp(R_{j^*},F_{j^*},\ldots,F(t))}(\Lambda)}{t}\right)\right]$$

Since $\theta_A^{exp(R_1,F_1,\ldots,R_{j^*-1},F_{j^*-1})}(\Lambda)$ is constant, in the limit, the first term is 0, and so we are interested in

$$\lim_{t\to\infty} u_i\left(\frac{\theta_a^{(R_{j^*},F_{j^*},\ldots,F(t))}(\Lambda)}{t}\right)$$

The expected frequency can be split up into the expected frequency for the sampling periods and for the free periods. Since Λ always determines that $\sigma_A^{\mathcal{M}}$ is not a correlated equilibrium, during all the free periods agents will always use L, and so we are interested in

$$\lim_{t\to\infty}\left[u_i\left(\frac{\theta_a^{(R_{j^*},\ldots,R(t))}(\Lambda)}{t}\right) + u_i\left(\frac{\theta_a^{(F_{j^*},\ldots,F(t))}(L)}{t}\right)\right]$$

Since we assumed that all utilities are nonnegative, we may discard the first term, and thus have

$$\lim_{t\to\infty} u_i\left(\frac{\theta_a^{(F_{j^*},\ldots,F(t)}(L))}{t}\right)$$

Therefore, by Lemma 1, the theorem follows. □

Together, Theorems 2 and 3 show that, with probability one, if $\sigma_A^{\mathcal{M}}$ is a correlated equilibrium, agents will converge to it and if $\sigma_A^{\mathcal{M}}$ is not a correlated equilibrium, agents will be no worse off in the long run for using Λ.

6 Conclusion

The setting for this paper was a repeated game with a mediator. The mediator makes suggestions to the agents as to what actions to take. We presented a test that agents could use so that, with high probability, they could determine if the mediator's suggestion was a correlated equilibrium. We then generalized our algorithm to incorporate repeated testing so that in the limit, with probability one, the test will always correctly determine whether the mediator's suggested strategy is a correlated equilibrium. As a result, if the mediator suggests a correlated equilibrium, then agents will converge to it, and otherwise, be no worse off in the long run for having used our algorithm.

We envision several directions for future research. First, it might be possible to extend our algorithm to work in radically uncoupled environments, where agents are not aware of the existence of others. This would significantly decrease the knowledge requirements of our test. Second, we would like to extend our approach so that the mediator receives feedback from the agents themselves, which can be used to help select appropriate correlated strategies. We believe that the incentive issues in such an approach will be challenging. It may also be interesting to apply our approach to other solution concepts such as mediated equilibria [11].

In a more applied direction, it might be possible to generalize our approach so it can be used in a stochastic game setting. Thus, our approach could be combined with methods such as Q-learning [7]. Correlated equilibria have also been used in graphical games, which can be used to model many different settings [10]. Hence, applying our technique to graphical games may yield some interesting results. For example, *network games* use graphical games to help represent a variety of problems, from public good provision and trade to information collection [6]. These models can be hindered by a "fundamental theoretical problem: even the simplest games played on networks have multiple equilibrium*[sic]* which display a bewildering range of possible outcomes" [6]. Our model may help integrate correlated equilibria as a possible solution to this problem.

7 Acknowledgements

Our thanks to Gord Hines for his statistical advice.

References

[1] R. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1:67–96, 1974.

[2] J. Cohen. *Statistical Power Analysis for the Behavioral Sciences*. 2nd edition, 1988.

[3] D. P. D. Farias and N. Megiddo. Combining expert advice in reactive environments. *Journal of the ACM*, 53(5):762–799, 2006.

[4] D. P. Foster and R. Vohra. Calibrated learning and correlated equilibrium. *Games and Economic Behavior*, 21:40–55, 1997.

[5] D. P. Foster and H. P. Young. Learning, hypothesis testing, and Nash equilibrium. *Games and Economic Behavior*, 45:73–96, 2003.

[6] A. Galeotti, S. Goyal, M. O. Jackson, F. Vega-Redondo, and L. Yariv. Network games. Unpublished, Jan 2006.

[7] A. Greenwald and K. Hall. Correlated Q-learning. In *Proceedings of ICML-2003*, pages 242–249, Washington, DC, USA, 2003.

[8] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68:1127–1150, 2000.

[9] N. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, volume 2. 1995.

[10] S. Kakade, M. Kearns, J. Langford, and L. Ortiz. Correlated equilibria in graphical games. In *EC '03: Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 42–47, New York, NY, USA, 2003.

[11] D. Monderer and M. Tennenholtz. Strong mediated equilibrium. In *Proceedings of the 21st American Association of Artificial Intelligence Conference*, Boston, MA, USA, 2006.

[12] L. Wasserman. *All of Statistics*. Springer, 2004.

A Proof of Lemma 1

Proof. Consider θ with respect to $a \in A$, denoted by θ_a. Since j^* is fixed, $\theta_a^{F_1,\ldots,F_{j^*-1}}(L)$ is constant, and therefore,

$$\lim_{t\to\infty} \frac{\theta_a^{exp(F_1,\ldots,F_{j^*-1})}(L)}{t} = 0, \tag{28}$$

and therefore, Equations 25 and 26 are equivalent.

Since the utility functions are linear transformations, proving the following is sufficient, although not necessary, to prove that Equation 25 holds,

$$\lim_{t\to\infty} \frac{\theta_a^{exp(t)}(L) - \theta_a^{exp(F_1,\ldots,F_{j(t)})}(L)}{t} = 0. \tag{29}$$

Since L is flexible, it will, in expectation, always behave the same way during each free period. Specifically,

$$\theta_a^{exp(b_{F_j}, b_{F_j}+l_{F_j})}(L) = \theta_a^{exp(b_{F_{j'}}, b_{F_{j'}}+l_{F_j})}(L), \tag{30}$$

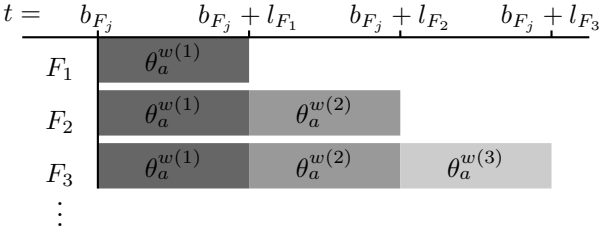


Figure 4: A graphical representation of how the expected frequency of play will be repeated each free period.

for all j' such that $l_{F_{j'}} \geq l_{F_j}$. This relationship can be represented graphically, as shown in Figure 4, where for simplicity, we let $w(j) = exp(b_{F_j} + l_{F_{j-1}}, b_{F_j} + l_{F_j})$, where $l_{F_0} = 0$. Therefore,

$$\theta_a^{exp(F_1,\ldots,F_{j(t)})}(L) = \sum_{j=1}^{j(t)} (j(t) - j + 1)\theta_a^{w(j)}(L).$$

Note that $\theta_a^{w(j)}$ will be "represented" more than $\theta_a^{w(j')}$ for $j < j'$ and any finite t. In order for Equation 29 to hold, in the limit, all $\theta_a^{w(j)}$ be must represented equally, *i.e.*

$$\lim_{t\to\infty} \frac{j(t) - j + 1}{t} = \lim_{t\to\infty} \frac{j(t) - j' + 1}{t}, \tag{31}$$

for all j, j'. Consider $t(j) = j^{-1}(t)$, *i.e.* the first time index after the j^{th} free period has ended:

$$t(j) = \sum_{j'=1}^{j} (l_{R_j} + l_{F_j}) \geq \sum_{j'=1}^{j} l_{R_j}. \tag{32}$$

By Equation 23, $\lim_{j\to\infty} \frac{t(j)}{j} = \infty$, and therefore,

$$\lim_{t\to\infty} \frac{j(t) - j + 1}{t} \leq \lim_{t\to\infty} \frac{j(t)}{t} = 0. \tag{33}$$

Therefore, in the limit, all $\theta_a^{w(j')}$ will be represented equally. However, since $\sum_{j=1}^{j(t)} l_{F_j} < t$, each $\theta_a^{w(j)}$ will be "underrepresented" compared to $\theta_a^t(L)$ for any finite t. However, in the limit, this is not the case since,

$$\begin{aligned}
\lim_{t\to\infty} \frac{\sum_{j=1}^{j(t)} l_{F_j}}{t} &= \lim_{t\to\infty} \frac{\sum_{j=1}^{j(t)} l_{F_j}}{\sum_{j=1}^{j(t)} (l_{R_j} + l_{F_j})} \\
&= \lim_{t\to\infty} \frac{1}{\frac{\sum_{j=1}^{j(t)} l_{R_j}}{\sum_{j=1}^{j(t)} l_{F_j}} + 1} \\
&= 1 \text{ (by Equation 22).}
\end{aligned} \tag{34}$$

Therefore, in the limit $\theta_a^{w(j)}$ will be represented equally compared to $\theta_a^t(L)$. □

Causal discovery of linear acyclic models with arbitrary distributions

Patrik O. Hoyer
Aapo Hyvärinen
Helsinki Institute for Information Technology &
Department of Computer Science
University of Helsinki
Finland

Richard Scheines
Peter Spirtes
Joseph Ramsey
Department of Philosophy
Carnegie Mellon University
Pittsburgh, PA, USA

Gustavo Lacerda
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA, USA

Shohei Shimizu
Osaka University
Japan

Abstract

An important task in data analysis is the discovery of causal relationships between observed variables. For continuous-valued data, linear acyclic causal models are commonly used to model the data-generating process, and the inference of such models is a well-studied problem. However, existing methods have significant limitations. Methods based on conditional independencies (Spirtes et al. 1993; Pearl 2000) cannot distinguish between independence-equivalent models, whereas approaches purely based on Independent Component Analysis (Shimizu et al. 2006) are inapplicable to data which is partially Gaussian. In this paper, we generalize and combine the two approaches, to yield a method able to learn the model structure in many cases for which the previous methods provide answers that are either incorrect or are not as informative as possible. We give exact graphical conditions for when two distinct models represent the same family of distributions, and empirically demonstrate the power of our method through thorough simulations.

1 INTRODUCTION

In much of science, the primary focus is on the discovery of *causal* relationships between quantities of interest. The randomized controlled experiment is geared specifically to inferring such relationships. Unfortunately, in many studies it is unethical, technically extremely difficult, or simply too expensive to conduct such experiments. In such cases causal discovery must be based on uncontrolled, purely observational data combined with prior information and reasonable assumptions.

In cases in which the observed data is continuous-valued, linear acyclic models (also known as recursive Structural Equation Models) have been widely used in a variety of fields such as econometrics, psychology, sociology, and biology; for some examples, see (Bollen 1989). In much of this work, the structure of the models has been assumed to be known or, at most, only a few different models have been compared. During the past 20 years, however, a number of methods have been developed to learn the model structure in an unsupervised way (Spirtes et al. 1993; Pearl 2000; Geiger and Heckerman 1994; Shimizu et al. 2006). Nevertheless, all approaches so far presented have either required distributional assumptions or have been overly restricted in the amount of structure they can infer from the data. In this contribution we show how to combine the strenghts of existing approaches, yielding a method capable of inferring the model structure in many cases where previous methods give incorrect or uninformative answers.

The paper is structured as follows: Section 2 precisely defines the models under study, and Section 3 discusses existing methods for causal discovery of such models. In Section 4 we formalize the discovery problem and give exact theoretical results on identifiability. Then, in Section 5 we introduce and analyze a method termed `PClingam` that combines the strenghts of existing methods and overcomes some of their weaknesses, and is, in the limit, able to estimate all identifiable aspects of the underlying model. Section 6 provides empirical demonstrations of the power of our method. Finally, Section 7 maps out future work and Section 8 provides a summary of the main points of the paper.

2 LINEAR MODELS

In this paper, we assume that the observed data has been generated by the following process:

1. The observed variables x_i, $i = \{1 \dots n\}$ can be arranged in a *causal order*, such that no later variable causes any earlier variable. We denote such a causal order by $k(i)$. That is, the generating process is *recursive* (Bollen 1989), meaning it can be represented graphically by a *directed acyclic graph* (DAG) (Pearl 2000; Spirtes et al. 1993).

2. The value assigned to each variable x_i is a *linear function* of the values already assigned to the earlier variables, plus a 'disturbance' (noise) term e_i, and plus an optional constant term c_i, that is

$$x_i = \sum_{k(j)<k(i)} b_{ij} x_j + e_i + c_i, \qquad (1)$$

   where we only include non-zero coefficients b_{ij} in the equation.

3. The disturbances e_i are all continuous random variables with arbitrary densities $p_i(e_i)$, and the e_i are independent of each other, i.e. $p(e_1, \dots, e_n) = \prod_i p_i(e_i)$.

This formulation neither requires the disturbances to be normally distributed nor does it require them to have non-Gaussian (non-normal) densities. In general, some of the distributions can be Gaussian and some not, and we do not a priori know which are which.

We assume that we are able to observe a large number of data vectors $\mathbf{x}$ (which contain the variables x_i), and each data vector is generated according to the above described process, with the same causal order $k(i)$, same coefficients b_{ij}, same constants c_i, and the disturbances e_i sampled independently from the same distributions. Note that the independence of the disturbances implies that there are *no unobserved confounders* (Pearl 2000). Spirtes et al. (1993) call this the *causally sufficient* case.

Finally, we assume that the observed distribution is *faithful* to the generating graph (Spirtes et al. 1993), i.e. the model is *stable* in the terminology of Pearl (2000). If the model parameters are in some sense randomly generated, this is not a strong assumption, as violations of faithfulness have Lebesgue measure 0 in the space of the linear coefficients.

An example of such a model is given in Figure 1a. Note that the full model consists of a directed acyclic graph over the variables, the connection strenghts b_{ij}, the constants c_i, and the densities $p_i(e_i)$. In this example we have chosen $c_i = 0$ for all i, so these are not shown.

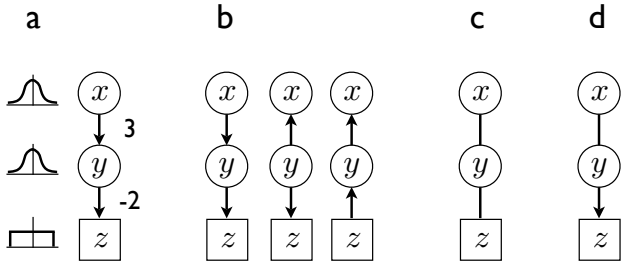


Figure 1: An example case used to illustrate the concepts described in Sections 2–4. **(a)** A linear, acyclic causal model for x, y and z. The data is generated as $x := e_x$, $y := 3x + e_y$, and $z := -2y + e_z$, with e_x and e_y drawn from Gaussian distributions and e_z from a non-Gaussian distribution, and e_x, e_y and e_z are all mutually independent. Note that we show variables with Gaussian disturbances using circles whereas variables with non-Gaussian disturbances are marked by squares. **(b)** The three directed acyclic graphs over x, y and z which all entail the same conditional independence relationships as the generating model. **(c)** The three DAGs in (b) succintly represented as a *d-separation-equivalence pattern*. **(d)** The *distribution-equivalence pattern* of the original model.

3 EXISTING METHODS

Given our set of data vectors $\mathbf{x}$, to what extent can we estimate the data generating process? Obviously, if the number N of data vectors is small, estimation may be quite unreliable. Therefore we will here mainly focus on the theoretical question: To what extent (and with what methods) can we identify the true model in the limit as $N \to \infty$?

The most well-known approach to inference of this type of causal networks is based on (conditional) independencies between the variables (Spirtes et al. 1993; Pearl 2000). When, as in our case, there are assumed to be no hidden confounding variables and no selection bias, one can in the large-sample limit identify the set of networks which represent the same independencies as the true data generating model. To illustrate, in Figure 1b we show all three DAGs which imply the set of independencies produced by the true model. This set is known as the d-separation-equivalence class, and is often represented in the form of a *d-separation-equivalence pattern*: a partly directed graph in which undirected edges represent edges for which both directions are present in the equivalence class (Spirtes et al. 1993), as illustrated in Figure 1c. We want to emphasize that, using conditional independence information alone, it is impossible to distinguish between members inside a d-separation-equivalence class because these (by definition) represent the same set of conditional

independencies between the observed variables.

Fortunately, in many cases there is additional information available that can be used to further distinguish between different DAGs. In particular, it can be shown (Shimizu et al. 2006) that if all (or all but one) distributions of the error variables are non-Gaussian, it is in fact possible to identify the complete causal model, including all the parameters. This is possible using a method based on Independent Component Analysis (ICA) (Hyvärinen et al. 2001). Unfortunately, however, when two or more disturbances are Gaussian the standard method based on ICA will fail. As an extreme example, when all disturbances are Gaussian, standard ICA-based methods return nonsense and are not even able to find the correct d-separation-equivalence class.

These considerations raise the question of whether it is possible to combine the methods so as to obtain robustness with respect to Gaussian distributions but not forgo the possibility of identifying the full model in favourable circumstances. Indeed, such a combination is possible and is presented in Section 5. Here, we simply note that the naïve solution of first running some test and then selecting one of the two methods, will not be optimal. Consider, for instance, our example model in Figure 1a. Because there is more than one Gaussian error variable the standard ICA-based method (Shimizu et al. 2006) is not applicable, and hence one would have to settle for the d-separation-equivalence class (Figure 1c) given by independence-based methods. However, as we show in the next section, in this example we can actually reject one of the DAGs in the equivalence class and hence obtain a smaller set of possible generating models.

4 DISTRIBUTION-EQUIVALENCE

First, we need to extend a DAG object to include information on the non-Gaussianity of associated disturbance variables.

Definition 1 *An* ***ngDAG*** *is a pair* (G, ng) *where* G *is a directed acyclic graph over a set of variables* V *and* ng *is a binary vector of length* $|V|$*, each element of which is associated with one of the variables of* V*.*

Definition 2 *We say that a linear acyclic causal model* M instantiates *an* ***ngDAG*** D *(alternatively,* D represents M*) if and only if the directed acyclic graph associated with* M *is equal to that specified in* D*, and further if the set of variables with non-Gaussian disturbance variables in* M *is equal to the set of positive entries in the binary vector specified in* D*.*

In general, an `ngDAG` D is instantiated by many different models M which differ in their connection strengths b_{ij} as well as in their distributions $p_i(e_i)$. Next, we define the important concept of distribution-equivalence between `ngDAG`s, which defines to what extent it is possible to infer the `ngDAG` which represents the true data generating causal model, from observational data alone.

Definition 3 *Two* ***ngDAGs*** D_1 *and* D_2 *are* distribution-equivalent *if and only if for any linear acyclic causal model* M_1 *which instantiates* D_1 *there exists an instantiation* M_2 *of* D_2 *which yields the same joint observed distribution as* M_1*, and vice versa.*

Distribution-equivalence partitions the set of `ngDAG`s into distribution-equivalence classes, and these may be represented using simplified graphs:

Definition 4 *An* ***ngDAG*** pattern *representing an* ***ngDAG*** D *is a mixed graph (consisting of potentially both directed und undirected edges), obtained in the following way:*

1. *Derive the d-separation-equivalence pattern corresponding to the DAG in* D
2. *Orient any unoriented edges which originate from, or terminate in, a node positively marked in* ng *of* D*, in the orientation given by the DAG in* D
3. *Finally, orient any edges which follow from the orientations given in the previous step and d-separation-equivalence, according to the rules derived by Meek (1995).*

We say that a mixed graph is an ***ngDAG*** *pattern if it represents* some ***ngDAG****.*

An `ngDAG` pattern is similar in many respects to d-separation-equivalence patterns. For example, we have the following result:

Lemma 1 *An* ***ngDAG*** *pattern is a chain graph.*

The proof is given in the Appendix.

Our main result connects `ngDAG` patterns with distribution-equivalence in mixed Gaussian and non-Gaussian models in the same way that d-separation-equivalence patterns are associated with distribution-equivalence in purely Gaussian models:

Theorem 1 *Two* ***ngDAGs*** *are distribution-equivalent if and only if they are represented by the same* ***ngDAG*** *pattern.*

The proof of this theorem is provided in the Appendix. The important point is that we now know exactly which models are indistinguishable from each other on the basis of observational data alone.

As a simple illustration, in Figure 1d we show the `ngDAG` pattern representing the `ngDAG` corresponding to the generating model of Figure 1a. Note that the `ngDAG` pattern is more informative than the d-separation-equivalence pattern of Figure 1c. Nevertheless, there are still two `ngDAG`s (leftmost two in Figure 1b) which cannot be distinguished based on non-experimental data.

Henceforth in the paper we shall use the terms *`ngDAG` pattern* and *distribution-equivalence pattern* interchangably.

5 PC-LINGAM

Although an important goal in this study was to look at the theoretical aspects of identifying DAGs in mixed Gaussian / non-Gaussian acyclic linear causal models, an equally significant objective is to give a practical method with which to infer models from a finite data set. Although there are a number of possible approaches, we here give a simple combination of independence-based techniques and the ICA-based method. The method, termed `PClingam`, consists of three steps:

1. Use methods based on conditional independence tests to estimate the d-separation-equivalence class within which the generating model lies. In particular, we advocate using the PC algorithm (Spirtes et al. 1993) which is computationally efficient even for a large number of variables. Note that, for linear models, to obtain the d-separation-equivalence class it is sufficient to identify the zero partial correlations in the data, as these depend only on the linear coefficients and the variances of the disturbances (and *not* on non-Gaussianity aspects of the distributions). However, since the data may well be signficantly non-Gaussian, non-parametric tests should optimally be used to find the zero partial correlations.

2. For each DAG G in the estimated d-separation-equivalence class:

   (a) Estimate the coefficients b_{ij} using ordinary least-squares regression. (Note that this provides consistent estimates regardless of non-Gaussianity of the variables.)

   (b) Calculate the corresponding residuals e_i and rescale them to zero mean and unit variance for each i

   (c) Calculate the corresponding ICA objective function

   $$U_f = \sum_i \left(E\{f(e_i)\} - k\right)^2 \qquad (2)$$

   where k is the expected value of f applied to a zero-mean, unit variance Gaussian variable, i.e. $k = E\{f(g)\},\ g \sim \mathcal{N}(0,1)$. In the ICA literature, many different choices of f have been utilized; here we suggest simply taking the absolute value function $f(e_i) = |e_i|$, giving

   $$U = \sum_i \left(E\{|e_i|\} - \sqrt{2/\pi}\right)^2 \qquad (3)$$

   Of course, since we only have samples we have to take the sample mean rather than the expectation.

3. Select the highest-scoring DAG G_{opt} from Step 2 and apply a statistical test for normality for each of the corresponding residuals e_i. Using Definition 4, compute and return the `ngDAG` pattern representing the `ngDAG` (G_{opt}, ng) where ng is the vector indicating those residuals whose normality was rejected by the normality tests.

The objective function U is commonly used in ICA as a measure of the non-Gaussianity of a random variable, and it can be shown to give a consistent estimator for finding independent components under weak conditions (Hyvärinen et al. 2001). ICA estimation is closely related to choosing the right DAG because statistical independence of the estimated residuals is a necessary condition for the correct model: Any DAG for which the estimated residuals are not independent violates the assumptions of the model (see Section 2) and hence cannot be the data-generating DAG. On the other hand, any DAG which results in statistically independent residuals represents one valid model that could have generated the data.

Note that if we could disregard sampling effects, distribution-equivalent models would attain exactly the same value of U. However, in the practical case of a finite sample this is not the case, thus Step 3 in the `PClingam` algorithm is required to identify the correct distribution-equivalence class.

The method as presented above has at least a couple of shortcomings. One is that, for any given function $f(e_i)$ used, there always exist distributions which are non-Gaussian yet are not distinguished from the Gaussian by this measure. This is a well-known issue in ICA which fortunately tends to have little practical significance since few such distributions are encountered in practice. If needed, non-parametric Gaussianity measures could be used to remedy this potential problem.

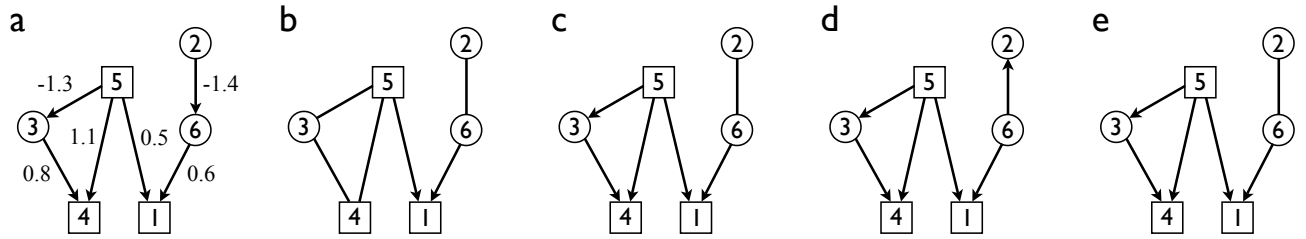


Figure 2: One of the networks used in the simulations. Variables with non-Gaussian disturbances are shown in squares, while those with Gaussian disturbances are plotted as circles. **(a)** True data-generating model. **(b)** True d-separation-equivalence pattern. **(c)** True distribution-equivalence pattern. **(d)** Estimated DAG G_{opt}. **(e)** Estimated distribution-equivalence pattern. See main text for details.

Naturally, in some cases many of the disturbances may be slightly non-Gaussian yet sufficiently close to Gaussian that the available samples may not be sufficient to distinguish the two and utilize the information for determining causal directions in the model. Of course, this is not a shortcoming of this particular method but is a more general phenomenon.

Another important limitation is that the ICA objective function given above will only provide a proper comparison of different DAGs for which the residuals e_i are linearly uncorrelated. This is guaranteed to be the case when the search is in the correct d-separation-equivalence class, but if in Step 1 of the procedure we select a too simple model (i.e. containing too few edges) then the estimated disturbances may be linearly correlated and the objective function misleading. Thus, it might be wise to include a term penalizing linear correlations such as is used in maximum likelihood estimation of ICA (Hyvärinen et al. 2001). However, to keep our method as simple as possible, we have omitted such a penalty term in this paper.

6 SIMULATIONS

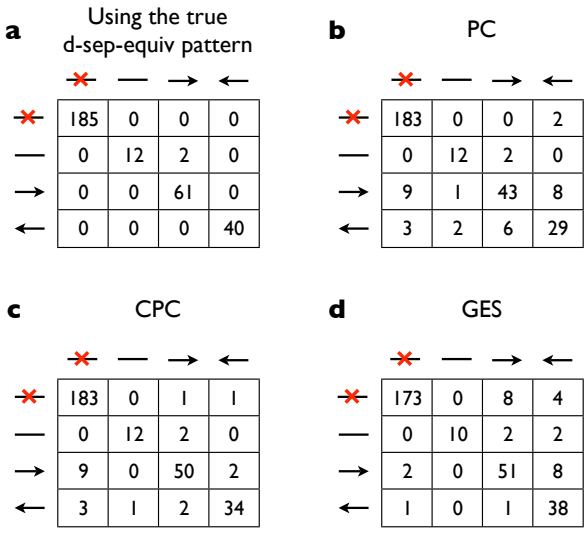
In this section we report on simulations used to test the performance of the `PClingam` method. First, we tested the ability of the non-Gaussianity objective function (3) of Step 2 and the normality tests of Step 3 of `PClingam` to identify the correct `ngDAG` pattern (distribution-equivalence class) when the true d-separation-equivalence pattern was known. In other words, we tested how well the algorithm would function if Step 1 of the method worked flawlessly. Subsequently, we experimented with the full method incorporating the necessary estimation of the d-separation-equivalence class (Step 1).

Figure 2a displays one of the models used to test the procedure. The disturbance distributions of variables X_1 and X_5 were a standard Gaussian the values of which were squared (but keeping the original sign) while the disturbance of X_4 was produced in a similar way but instead raising the values to the third power. The disturbances of X_2, X_3, and X_6 were Gaussian. The disturbance variables were scaled such that their variances ranged from 1.0 to 3.0. A sample of 1000 data vectors was generated from the model.

Figure 2b shows the true d-separation-equivalence pattern of the model in (a). The equivalence class consists of 12 different DAGs. However, the non-Gaussianity of the disturbances of X_1, X_4, and X_5 means that there are actually only 2 DAGs which are distribution-equivalent; these are represented by the distribution-equivalence pattern of Figure 2c. Figure 2d shows the DAG G_{opt} found by Step 2 of `PClingam` from the data, when the true d-separation-equivalence class was given to the algorithm. An Anderson-Darling test for normality (Anderson and Darling 1954) gave the p-values 0.000, 0.3145, 0.2181, 0.000, 0.000, and 0.0197 for the corresponding residuals e_1 to e_6. Inferring a residual to be non-Gaussian when $p < 0.01$ in Step 3 of the method produced the `ngDAG` pattern of Figure 2e, which turns out identical to the true `ngDAG` pattern in (c).

This basic procedure was repeated 20 times, with the results summarized in Table 1a. In each simulation, we randomly generated a linear acyclic causal model over 6 variables, with each variable randomly chosen to have either a Gaussian or a non-Gaussian disturbance. The non-Gaussian distributions used were those mentioned above as well as a Student's t (2 degrees of freedom), a bimodal Mixture of Gaussians $(0.5\mathcal{N}(-2,1) + 0.5\mathcal{N}(2,1))$, a log-normal distribution (exponentiated standard normal) and a uniform distribution. The true d-separation-equivalence pattern was input to the algorithm, to test the functioning of the `PClingam` method when the correct pattern is selected in Step 1. The panel shows how often a specific type of true edge (in the true distribution-equivalence pattern) gave rise to a specific type of estimated edge (in the estimated distribution-equivalence pattern). Rows cor-

Table 1: Summary of the simulations employing various methods for inferring the d-separation-equivalence class in Step 1 of `PClingam`. Each table is a confusion matrix of arcs in the true distribution-equivalence patterns vs arcs in the estimated distribution-equivalence pattern. See main text for details.



a	✱←	—	→	←
✱←	185	0	0	0
—	0	12	2	0
→	0	0	61	0
←	0	0	0	40

b	✱←	—	→	←
✱←	183	0	0	2
—	0	12	2	0
→	9	1	43	8
←	3	2	6	29

c	✱←	—	→	←
✱←	183	0	1	1
—	0	12	2	0
→	9	0	50	2
←	3	1	2	34

d	✱←	—	→	←
✱←	173	0	8	4
—	0	10	2	2
→	2	0	51	8
←	1	0	1	38

respond to the true edges, columns to estimated ones. Optimally all off-diagonal elements would be zero. It can be seen that the results are close to perfect; the method misclassifies two undirected edges as directed, but correctly estimates all others.

These simulations confirm that the `PClingam` method works well at least when the d-separation-equivalence class can reliably be estimated. But in practice, with finite datasets, there may be significant errors in inferring the d-separation-equivalence class. The degree to which this affects the algorithm is an important practical issue.

Thus, in further simulations, we applied several different methods for learning d-separation-equivalence patterns from the simulated data, as Step 1 in the `PClingam` method. The methods we compared were the PC algorithm (Spirtes et al. 1993), the Conservative PC algorithm (Ramsey et al. 2006), and the GES algorithm (Chickering 2002). Panels b-d of Table 1 summarize the results. Although all of the methods assumed Gaussianity when learning the d-separation-equivalence pattern, the results are still quite encouraging, and a clear majority of edges were correctly estimated.

7 FUTURE WORK

While the theoretical aspects of identifiability are solved, at least a couple of important issues regarding the estimation of the model from finite samples remain.

First and foremost, non-parametric methods for identifying zero partial correlations in non-Gaussian settings should be used so as to obtain better estimates of the appropriate d-separation-equivalence class within which to search. Although the methods developed for Gaussian variables seem to work relatively well in our partly non-Gaussian setting, it is likely they will be outperformed by methods that take into account the possibility of non-Gaussian distributions.

Another important question is how to make the procedure scalable to data involving many (tens or even hundreds of) variables. Although the current approach relies on a brute-force enumeration of all DAGs in the d-separation-equivalence class, it would not be difficult to adapt the method to do a local search among DAGs in an equivalence class. The extent to which such a method would be hampered by local maxima is unknown.

8 SUMMARY

The discovery of linear acyclic causal models is a topic which has been thoroughly investigated in the last two decades. Both the Gaussian and the fully non-Gaussian special cases are well understood, but the general mixed case has not been previously discussed. In this paper we have provided a complete characterization of distribution-equivalence and a practical estimation method in this setting.

Acknowledgements

The authors wish to thank Clark Glymour for helpful and stimulating discussions. P.O.H. was funded by a postdoctoral researcher grant from the University of Helsinki.

References

Anderson, T. W. and D. A. Darling (1954). A test of goodness of fit. *Journal of the American Statistical Association 49*(268), 765–769.

Andersson, S. A., D. Madigan, and M. D. Perlman (1997). A characterization of markov equivalence classes for acyclic digraphs. *Annals of Statistics 25*, 505–541.

Bollen, K. A. (1989). *Structural Equations with Latent Variables*. John Wiley & Sons.

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of Machine Learning Research 3*, 507–554.

Geiger, D. and D. Heckerman (1994). Learning gaussian networks. In *Proc. of the 10th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 235–243.

Hyvärinen, A., J. Karhunen, and E. Oja (2001). *Independent Component Analysis*. Wiley.

Meek, C. (1995). Causal inference and causal explanation with background knowledge. In *Proc. of the 11th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 403–411.

Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

Ramsey, J., J. Zhang, and P. Spirtes (2006). Adjacency-faithfulness and conservative causal inference. In *Proc. of the 22nd Annual Conference on Uncertainty in Artificial Intelligence*.

Shimizu, S., P. O. Hoyer, A. Hyvärinen, and A. J. Kerminen (2006). A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research 7*, 2003–2030.

Spirtes, P., C. Glymour, and R. Scheines (1993). *Causation, Prediction, and Search*. Springer-Verlag. (2nd ed. MIT Press 2000).

APPENDIX

Proof of Lemma 1: Select any `ngDAG` D represented by the `ngDAG` pattern P in question. Now, for each variable X_i in D with a non-Gaussian disturbance, add two auxiliary variables both of which have no parents and both of which have X_i as their only child. Now, consider what the d-separation-equivalence pattern looks like for this *augmented* graph. Any edge in D *into* such an X_i may be oriented in the d-separation-equivalence pattern on the basis of the resulting unshielded collider at X_i. Furthermore, any edge in D *out of* such an X_i may similarly be oriented on the basis of the lack of an unshielded collider. Thus, all edges either originating from, or terminating in, a variable with a non-Gaussian disturbance will be oriented in the d-separation-equivalence pattern of the augmented graph. Additionally, any edges whose orientations can be deduced as a result of knowing the newly oriented edges are oriented as well. Hence, adding the auxiliary variables has exactly the same effect as simply orienting any edges connected to a non-Gaussian variable.

Because d-separation-equivalence patterns are always chain graphs (Andersson et al. 1997), the d-separation-equivalence pattern for the augmented graph is a chain graph. Since each of the auxiliary variables is connected to the rest of the graph only by a single oriented edge, removing these variables cannot change the chain graph property of the graph. Hence, the distribution-equivalence pattern obtained by orienting any edges originating from, or terminating in, variables with non-Gaussian variables (and subsequently orienting any edges deduced from these) has to be a chain graph. This completes the proof.

Proof of Theorem 1: We will first prove that if two `ngDAG`s D_1 and D_2 are distribution-equivalent then they must share the same `ngDAG` pattern. Since distribution-equivalence implies d-separation-equivalence it is clear they have to share the d-separation-equivalence pattern. Thus we need to show that it is always possible to correctly orient any edges directly connected to any variable with a non-Gaussian disturbance.

Consider a linear acyclic causal model M_1 which instantiates D_1. Let us for simplicity of notation assume that the variables have been named $X_1, \ldots, X_n$ such that $j < i$ whenever X_j is an ancestor of X_i. Then it is possible to collect the linear coefficients b_{ij} which represent the direct causal effects into a *lower-triangular* matrix $\mathbf{B}$ such that we have $\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{e}$, where $\mathbf{x}$ collects the observed variables $X_1, \ldots, X_n$ and $\mathbf{e}$ denotes the disturbance variables. Note that we have for simplicity of notation and without loss of generality assumed the constants in (1) are equal to zero. Solving for $\mathbf{x}$ we obtain $\mathbf{x} = \mathbf{A}\mathbf{e}$ where $\mathbf{A} = (\mathbf{I} - \mathbf{B})^{-1}$ is the *reduced-form* matrix (representing the total effects between the variables), which here is lower-triangular due to the above causal ordering of the variables. By the assumptions of the model, the components of $\mathbf{e}$ are mutually independent. This means that our generating model is an ICA model, although several (potentially even all) of the components may be Gaussian. Although it is impossible to completely estimate $\mathbf{A}$ when there are two or more Gaussian components, it is well known (Hyvärinen et al. 2001) that the basis vectors (columns of $\mathbf{A}$) corresponding to the non-Gaussian components are identifiable (except for the standard indeterminancy of permutation and scaling), as is the covariance matrix of the (generally multidimensional) Gaussian component which groups all Gaussian disturbances.

Thus each non-Gaussian disturbance in the generating model essentially gives us a vector $\mathbf{a}_i$ containing the total effects of its corresponding variable X_i on the other variables. As no variable can affect its non-descendants, and as we have assumed faithfulness, the set of non-zero entries of each such vector exactly represents the union of the correct corresponding variable and all its descendants. If we could somehow know to which observed variable X_i each non-Gaussian basis

vector $\mathbf{a}_i$ should be paired we could easily orient all edges connecting to that variable as follows: Any observed variable X_j which we know (by d-separation) to be connected to X_i but which has a zero entry in $\mathbf{a}_i$ has to be a parent of X_i. Similarly, any observed variable X_j which we know (by d-separation) to be connected to X_i but which has a non-zero entry in $\mathbf{a}_i$ has to be a child of X_i.

But is it always possible to identify to which observed variable a given basis vector should be connected? We now show that this is indeed so. Consider what happens if we try to build a model in which a basis vector $\mathbf{a}_i$ is paired with a descendant X_j of X_i, rather than with the correct choice X_i. Because we are restricted to acyclic models, the new model may be represented by a reduced-form matrix $\mathbf{A'}$ which is lower-triangular for the new variable order (but not for the old order, because in this new model X_i is necessarily a descendant of X_j). Note also that, when the rows are identically ordered, the column $\mathbf{a}_i$ must be equal to column $\mathbf{a}'_j$ of $\mathbf{A'}$. Now, in the new model, X_i has to be represented by some disturbance variable, so there must be a column $\mathbf{a}'_i$ of $\mathbf{A'}$ which has a non-zero entry in the row corresponding to X_i. Furthermore, $\mathbf{a}'_i$ must have zeros wherever $\mathbf{a}_i$ has zeros. These properties imply that $\mathbf{a}'_i$ cannot be expressed as a linear combination of the $\{\mathbf{a}_k\}_{k\neq i}$: Since $\mathbf{a}_i$ cannot be used, one would have to depend on $\mathbf{a}_k$ corresponding to X_k which are ancestors of X_i to properly represent the non-zero entry corresponding to X_i in $\mathbf{a}'_i$. But among these there necessarily has to be an $\mathbf{a}_k$ which corresponds to a (relative) source X_k whose effect cannot be cancelled out by the others to lead to a required zero in $\mathbf{a}'_i$. Thus $\mathbf{a}'_i$ is not in $\mathrm{span}(\{\mathbf{a}_k\}_{k\neq i})$. This implies that $\mathrm{span}(\{\mathbf{a}_k\}_{k\neq i}) \neq \mathrm{span}(\{\mathbf{a}'_k\}_{k\neq j})$ and hence the covariance implied by $\mathbf{A'}$ cannot equal that implied by $\mathbf{A}$ and the two models cannot be distribution-equivalent. Hence to represent the correct distribution one has to pair basis vectors corresponding to non-Gaussian components to the correct observed variable, allowing the identification of the orientation of all edges directly connected to that observed variable.

Next we need to prove that two `ngDAG`s which share the same `ngDAG` pattern are distribution-equivalent. Call the two `ngDAG`s D_1 and D_2. Since they are represented by the same `ngDAG` pattern they may only differ in terms of the orientations of arrows within each chain component of the chain graph that is the `ngDAG` pattern. (A chain component is a maximal connected set of nodes such that there is no oriented edge between any pair of nodes.) Let V_c stand for the set of variables (necessarily all with Gaussian disturbances) making up any given chain component. First, note that we can focus our attention on only the variables V_c and any immediate parents of any of these: If we can match any conditional distribution $P(V_c \mid \mathrm{pa}(V_c))$ of a model M_1 represented by D_1 using a model M_2 represented by D_2 then we can always match the full joint distribution over all variables. Next consider the parents of the chain component. It is clear that each such parent must be connected to *all* the variables of the chain component, otherwise some edges in the component could be oriented using d-separation-equivalence.

Select any parametrization M_1 of D_1, and call the resulting reduced-form matrix *between the variables V_c only* $\mathbf{A}_1$. Because of the old result that d-separation-equivalence among Gaussian variables implies distribution-equivalence we know that there is a parametrization M_2 of D_2 such that the corresponding reduced-form matrix is $\mathbf{A}_2$, and we have $\mathbf{A}_1\mathbf{A}_1^T = \mathbf{A}_2\mathbf{A}_2^T$. Now consider the effect of a parent. Denoting the weigths of the parent onto the variables V_c, in model M_1, by the vector $\mathbf{w}_1$, we have $V_c = \mathbf{A}_1(\mathbf{e}+\mathbf{w}_1 p)$ where $\mathbf{e}$ represents the independent Gaussian disturbances of the V_c and p the value of the parent. Thus the conditional distribution of the V_c given the value p of the parent is a Gaussian with mean vector $\mathbf{A}_1\mathbf{w}_1 p$ and covariance matrix $\mathbf{A}_1\mathbf{A}_1^T$. For model M_2 we similarly have a Gaussian with mean $\mathbf{A}_2\mathbf{w}_2 p$ and covariance matrix $\mathbf{A}_2\mathbf{A}_2^T$. As noted above, because of d-separation-equivalence, these covariance matrices can be made identical with a suitable choice of $\mathbf{A}_2$. Then, selecting $\mathbf{w}_2 = \mathbf{A}_2^{-1}\mathbf{A}_1\mathbf{w}_1$ yields identical means as well. (Note that the reduced-form matrices are always invertible.). Also note that additional parents always add independently, and the weights can be selected in the above fashion for each parent separately. Thus, in summary, for any chain component on which models D_1 and D_2 have conflicting edge orientations, for any parametrization M_1 of D_1 it is always possible to find a parametrization M_2 of D_2 such that the conditional distribution of the chain component given its parents is identical. Thus the two models represent the exact same set of joint distributions over all the variables. This concludes the proof of the theorem.

Cumulative distribution networks and the derivative-sum-product algorithm

Jim C. Huang and Brendan J. Frey
Probabilistic and Statistical Inference Group, University of Toronto
10 King's College Road, Toronto, ON
M5S 3G4, Canada

Abstract

We introduce a new type of graphical model called a 'cumulative distribution network' (CDN), which expresses a joint cumulative distribution as a product of local functions. Each local function can be viewed as providing evidence about possible orderings, or rankings, of variables. Interestingly, we find that the conditional independence properties of CDNs are quite different from other graphical models. We also describe a message-passing algorithm that efficiently computes conditional cumulative distributions. Due to the unique independence properties of the CDN, these messages do not in general have a one-to-one correspondence with messages exchanged in standard algorithms, such as belief propagation. We demonstrate the application of CDNs for structured ranking learning using a previously-studied multi-player gaming dataset.

1 Introduction

Probabilistic graphical models are widely used for compactly representing joint probability density functions (PDFs)[1]. While such models have been successfully applied to a variety of problems, there are many tasks in which the joint probability density does not arise naturally and other representations may be more appropriate. In particular, the *cumulative distribution function* (CDF) is a probabilistic representation which arises frequently in a wide variety of applications such as ranking learning [9], survival analysis and data censuring [3]. In the setting of ranking learning, the goal is to model an ordinal variable y given an input set of features $\mathbf{x}$ while accounting for noise in the ranking process. The conditional CDF here accounts for the ordinal nature of y in addition to model uncertainty.

[1]We use PDF in reference to either the probability density function in the case of continuous random variables or the probability mass function in the case of discrete random variables

For problems of ranking learning that exhibit structure in the form of predicting multiple ordinal variables, a more flexible representation is required. Examples of this type of problem include predicting movie ratings for multiple movies or predicting multiplayer game outcomes with a team structure [4]. In such settings, we may wish to model not only stochastic ordering relationships between variable states, but also preferences between model variables and stochastic independence relationships between variables. This requires representing the joint CDF of many variables whilst explicitly accounting for both marginal and conditional independencies between variables.

Motivated by the above two problems, we present the cumulative distribution network (CDN), a novel graphical model which describes the joint CDF of a set of variables instead of the joint PDF. We show that CDNs provide a compact way to represent stochastic ordering relationships amongst variables and that the conditional independence properties of CDNs are quite different from previously studied graphical models describing PDFs (Bayesian networks [12], Markov random fields [6], factor graphs [7] and chain graphs [1]). In contrast to those, marginalization in CDNs involves tractable operations such as computing derivatives of local functions. We derive relevant theorems and lemmas for CDNs and describe a message-passing algorithm called the *derivative-sum-product algorithm* (DSP) for performing inference in such models. Finally we present results on an application to structured ranking learning in a multiplayer online game setting.

1.1 Example: Expressing conditional dependencies between variables

To illustrate a situation in which we have a set of conditional dependence relationships that cannot be represented by either a Markov random field or a Bayesian network, consider the following probability model with 4 binary variables X_1, X_2, X_3, X_4.

From the above probability model, we can establish

x_1	x_2	x_3	x_4	$P(x_1, x_2, x_3, x_4)$
0	0	0	0	343/1800
0	0	0	1	392/1800
0	0	1	0	105/1800
0	0	1	1	168/1800
0	1	0	0	105/1800
0	1	0	1	120/1800
0	1	1	0	87/1800
0	1	1	1	120/1800
1	0	0	0	49/1800
1	0	0	1	56/1800
1	0	1	0	15/1800
1	0	1	1	24/1800
1	1	0	0	63/1800
1	1	0	1	72/1800
1	1	1	0	33/1800
1	1	1	1	48/1800

Table 1: Example probability model over 4 binary variables X_1, X_2, X_3, X_4.

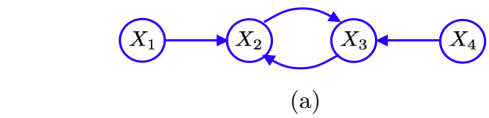
that $X_1 \not\perp\!\!\!\perp X_3 | X_2$ (i.e.: X_1 is dependent of X_3 given X_2), as $P(x_1, x_3|x_2) \neq P(x_1|x_2)P(x_3|x_2)$:

	$P(x_1, x_3\|x_2)$		$P(x_1\|x_2)P(x_3\|x_2)$	
x_1, x_3	$x_2 = 0$	$x_2 = 1$	$x_2 = 0$	$x_2 = 1$
0,0	245/384	75/216	245/384	80/216
0,1	91/384	69/216	91/384	64/216
1,0	35/384	45/216	35/384	40/216
1,1	13/384	27/216	13/384	32/216

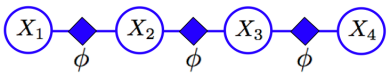
One can also verify that $X_2 \not\perp\!\!\!\perp X_4 | X_3$, $X_1 \not\perp\!\!\!\perp X_2$, $X_2 \not\perp\!\!\!\perp X_3$, $X_3 \not\perp\!\!\!\perp X_4, X_1 \perp\!\!\!\perp X_4, X_1 \perp\!\!\!\perp X_3, X_2 \perp\!\!\!\perp X_4$ from the above joint probability. In fact, we cannot represent the above set of conditional independence and dependence relationships using either a Markov random field, a Bayesian network or a factor graph, as $X_1 \not\perp\!\!\!\perp X_3 | X_2$ and $X_1 \perp\!\!\!\perp X_3$ cannot be simultaneously satisfied under such frameworks (similarly for $X_2 \not\perp\!\!\!\perp X_4 | X_3$ and $X_2 \perp\!\!\!\perp X_4$). Attempting to construct a directed model would lead to the introduction of directed cycles (Figure 1(a)) and would not be valid as a Bayesian network. However, the probability model in this example can be represented in compact form as a CDN (Figure 1(b)), which we will now proceed to define.

2 Cumulative distribution networks (CDNs)

Let $\mathbf{X} = \{X_1, \cdots, X_K\}$ denote a collection of K real-valued ordinal random variables. Let $\mathbf{x} = (x_1, \cdots, x_K)$ denote a vector of assignments for these variables under a joint PDF given by $P(\mathbf{x}) = P(x_1, \cdots, x_K) = Pr\{X_1 = x_1, \cdots, X_K = x_K\}$. Similarly, let $\mathbf{X} \setminus X_k$ denote the set $\mathbf{X}$ with X_k deleted. Let $\mathbf{X}_s$ denote some subset of the set of variables $\mathbf{X}$. We will often resort to the differentiation or finite difference operator with respect to a set of variables $\mathbf{X}_s$: we will use the notation $\partial_{\mathbf{x}_s}\Big[\cdot\Big]$ to denote this.



(a)



(b)

x	y	$\phi(x, y)$
0	0	$\frac{7}{10}$
0	1	$\frac{8}{10}$
1	0	$\frac{8}{10}$
1	1	1

(c)

Figure 1: a) Attempting to construct a directed model that encodes all conditonal independence and dependence relationships in the model of Table 1 leads to directed cycles, which are improper; b) The corresponding cumulative distribution network (CDN) over the 4 variables X_1, X_2, X_3, X_4; c) Function $\phi(x, y)$ used to construct the CDN in Table 1.

Our analysis here assumes the existence of the joint PDF $P(\mathbf{x})$ over a set of variables. The joint CDF $F(\mathbf{x}) = F(x_1, \cdots, x_K) = Pr\{X_1 \leq x_1, \cdots, X_K \leq x_K\}$ is defined as a function over $\{x_1, \cdots, x_K\}$ such that $\partial_{\mathbf{x}}\Big[F(\mathbf{x})\Big] = P(\mathbf{x})$ and

$$\sup_{\mathbf{x}} F(\mathbf{x}) = 1 \tag{1}$$

$$\inf_{x_k} F(x_k, \mathbf{x} \setminus x_k) = 0 \quad \forall k = 1, \cdots, K \tag{2}$$

$$\partial_{\mathbf{x}_s}\Big[F(\mathbf{x}_s, \mathbf{x} \setminus \mathbf{x}_s)\Big] \geq 0 \quad \forall \mathbf{X}_s \subseteq \mathbf{X} \tag{3}$$

We will now define the concept of a cumulative distribution network as a graphical model that efficiently expresses a CDF as a product of local functions, each of which captures a local cumulative distribution function.

Definition. *For K random variables $X_1, \cdots, X_K$, the cumulative distribution network (CDN) is an undirected bipartite graphical model consisting of a set of variable nodes corresponding to variables in $\mathbf{X}$, cumulative function nodes defined over subsets of these variables, a set of undirected edges linking variables to functions and a specification for each cumulative function. More precisely, let $\mathcal{C}$ denote the set of cumulative functions so that for $c \in \mathcal{C}$, $\phi_c(\mathbf{x}_c)$ is a cumulative function defined over neighboring variables $\mathbf{X}_c$. Let $\mathrm{N}(X_i) = \{c \in \mathcal{C} | X_i \in \mathbf{X}_c\}$ denote the set of all neighboring functions for variable X_i and let $\mathrm{N}(\phi_c) = \mathbf{X}_c$ be the set of variable nodes that ϕ_c is connected to. Simi-*

larly, let $\mathrm{N}(\mathbf{X}_s) = \bigcup_{X_i \in \mathbf{X}_s} \mathrm{N}(X_i)$ *denote the set of all neighboring functions for variable set* $\mathbf{X}_s$*. For a CDN with* K *variable nodes, the joint CDF of* $X_1, \cdots, X_K$ *is given by*

$$F(\mathbf{x}) = F(x_1, \cdots, x_K) = \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c). \tag{4}$$

As an example CDN, consider a joint CDF over 3 variables X, Y, Z which can be expressed as

$$F(x, y, z) = \phi_a(x, y)\phi_b(x, y, z)\phi_c(y, z)\phi_d(z). \tag{5}$$

This can be represented using the undirected graph shown in Figure 2, where each function node corresponds to one of the functions $\phi_c(\mathbf{x}_c)$.

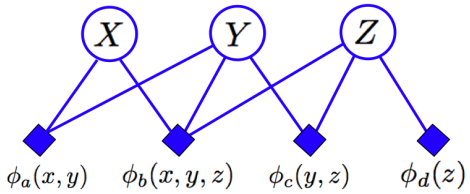


Figure 2: A cumulative distribution network (CDN) over 3 variables and 4 functions.

In order for the CDN to describe a joint CDF that corresponds to a valid PDF, we require that each of the cumulative functions satisfy $\phi_c(\mathbf{x}_c) \geq 0$. Furthermore, we will assume in the sequel that the derivatives/finite differences of each $\phi_c(\mathbf{x}_c)$ with respect to all of its argument variables can be properly computed. Finally, to satisfy the basic properties (1) to (3), the functions $\phi_c(\mathbf{x}_c)$ must satisfy the following conditions.

Positive convergence For all $c \in \mathcal{C}$, (1) holds if

$$\sup_{\mathbf{x}_c} \phi_c(\mathbf{x}_c) = 1. \tag{6}$$

We note that the former condition (6) is a sufficient, but not a necessary condition for (1) to hold, as we could allow for functions to converge to different limits whilst retaining the property that $\sup_{\mathbf{x}} F(\mathbf{x}) = 1$.

Negative convergence For any $X_i \in \mathbf{X}$, (2) holds if and only if at least one of its neighboring functions $\phi_c(\mathbf{x}_c)$ goes to zero as x_i goes to zero, i.e.

$$\forall X_i, \exists c \in \mathrm{N}(X_i) \,|\, \inf_{x_i} \phi_c(\mathbf{x}_c) = 0. \tag{7}$$

This can be proven as follows. For any $X_i \in \mathbf{X}$, $c \in \mathrm{N}(X_i)$, if $\inf_{x_i} \phi_c(\mathbf{x}_c) = 0$, then $\inf_{x_i} F(\mathbf{x}) = 0$, as

$$\begin{aligned}
\inf_{x_i} F(\mathbf{x}) &= \inf_{x_i} \prod_{c \in \mathrm{N}(X_i)} \phi_c(x_i, \mathbf{x}_c \setminus x_i) \prod_{c \notin \mathrm{N}(X_i)} \phi_c(\mathbf{x}_c) \\
&= \prod_{c \notin \mathrm{N}(X_i)} \phi_c(\mathbf{x}_c) \prod_{c \in \mathrm{N}(X_i)} \inf_{x_i} \phi_c(x_i, \mathbf{x}_c \setminus x_i) \\
&= 0.
\end{aligned}$$

Conversely,

$$\begin{aligned}
&\inf_{x_i} F(\mathbf{x}) = \inf_{x_i} \prod_{c \in \mathrm{N}(X_i)} \phi_c(x_i, \mathbf{x}_c \setminus x_i) \prod_{c \notin \mathrm{N}(X_i)} \phi_c(\mathbf{x}_c) = 0 \\
&\quad \Rightarrow \exists c \in \mathrm{N}(X_i) | \inf_{x_i} \phi_c(\mathbf{x}_c) = 0.
\end{aligned}$$

Monotonicity For any $\mathbf{x}$ such that $F(\mathbf{x}) > 0$ and for any $X_i \in \mathbf{X}$, $\partial_{x_i}\Big[F(\mathbf{x})\Big] \geq 0$ holds if and only if

$$\sum_{c \in \mathrm{N}(X_i)} \frac{\partial_{x_i}\Big[\phi_c(\mathbf{x}_c)\Big]}{\phi_c(\mathbf{x}_c)} \geq 0. \tag{8}$$

To prove this, since $F(\mathbf{x}) = \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c)$, we have

$$\begin{aligned}
\partial_{x_i}\Big[F(\mathbf{x})\Big] &= \partial_{x_i}\Big[\prod_{c \in \mathrm{N}(X_i)} \phi_c(\mathbf{x}_c) \prod_{c \notin \mathrm{N}(X_i)} \phi_c(\mathbf{x}_c)\Big] \\
&= \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c) \sum_{c \in \mathrm{N}(X_i)} \frac{1}{\phi_c(\mathbf{x}_c)} \partial_{x_i}\Big[\phi_c(\mathbf{x}_c)\Big] \\
&= F(\mathbf{x}) \sum_{c \in \mathrm{N}(X_i)} \frac{1}{\phi_c(\mathbf{x}_c)} \partial_{x_i}\Big[\phi_c(\mathbf{x}_c)\Big] \geq 0 \\
&\Rightarrow \sum_{c \in \mathrm{N}(X_i)} \frac{\partial_{x_i}\Big[\phi_c(\mathbf{x}_c)\Big]}{\phi_c(\mathbf{x}_c)} \geq 0.
\end{aligned}$$

Conversely,

$$\begin{aligned}
&\sum_{c \in \mathrm{N}(X_i)} \frac{1}{\phi_c(\mathbf{x}_c)} \partial_{x_i}\Big[\phi_c(\mathbf{x}_c)\Big] = \partial_{x_i}\Big[\log \prod_{c \in \mathrm{N}(X_i)} \phi_c(\mathbf{x}_c)\Big] \\
&= \partial_{x_i}\Big[\log \frac{F(\mathbf{x})}{\prod_{c \notin \mathrm{N}(X_i)} \phi_c(\mathbf{x}_c)}\Big] = \partial_{x_i}\Big[\log F(\mathbf{x})\Big] \\
&= \frac{1}{F(\mathbf{x})} \partial_{x_i}\Big[F(\mathbf{x})\Big] \geq 0 \Rightarrow \partial_{x_i}\Big[F(\mathbf{x})\Big] \geq 0.
\end{aligned}$$

The above result points to a sufficient condition for the CDN to correspond to a valid PDF, as demonstrated by the following lemma.

Lemma For all $\phi_c, X_i \in \mathrm{N}(\phi_c)$, $\partial_{x_i}\Big[F(\mathbf{x})\Big] \geq 0$ if $\partial_{x_i}\Big[\phi_c(\mathbf{x}_c)\Big] \geq 0$. To prove this, since $\phi_c(\mathbf{x}_c) \geq 0 \, \forall c \in \mathcal{C}$, we see that $\partial_{x_i}\Big[\phi_c(\mathbf{x}_c)\Big] \geq 0$ immediately satisfies $\sum_{c \in \mathrm{N}(X_i)} \frac{\partial_{x_i}\Big[\phi_c(\mathbf{x}_c)\Big]}{\phi_c(\mathbf{x}_c)} \geq 0$.

More generally, the above also holds for derivatives with respect to subsets of variables $\mathbf{x}_s$. That is, for all $c \in \mathcal{C}$, $\mathbf{X}_c = \mathrm{N}(\phi_c)$ and all possible subsets of variables $\mathbf{X}_s \subseteq \mathbf{X}_c$ such that $F(\mathbf{x}_s, \mathbf{x} \setminus \mathbf{x}_s)$ is strictly positive, $\partial_{\mathbf{x}}\Big[F(\mathbf{x})\Big] \geq 0$ holds if $\partial_{\mathbf{x}_s}\Big[\phi_c(\mathbf{x}_c)\Big] \geq 0$. This can be proven by noting that computing higher-order derivatives of $F(\mathbf{x})$ yields sums over products of derivatives,

so if $\phi_c(\mathbf{x}_c)$ and all higher-order derivatives with respect to each of their arguments are non-negative, then the resulting joint CDF must be monotonically non-decreasing and so (3) is satisfied.

3 Marginal and conditional independence properties of CDNs

In this section, we will highlight the marginal and conditional independence properties of CDNs, the latter of which significantly differ from those of graphical density models such as Bayesian networks, Markov random fields and factor graphs.

Theorem (Marginal Independence). *Two disjoint sets of variables $\mathbf{X}_s$ and $\mathbf{X}_t$ are marginally independent if there are no cumulative functions $c \in \mathcal{C}$ connecting variable nodes in $\mathbf{X}_s$ and $\mathbf{X}_t$.*

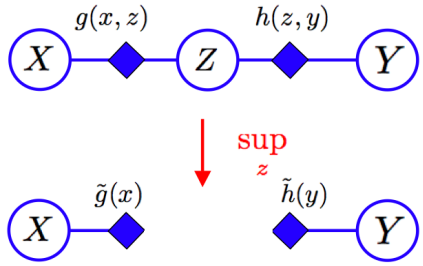


Figure 3: Marginal independence property of CDNs: unless two variables X and Y are connected to a common function node, they are marginally independent.

Proof. Intuitively, $\mathbf{X}_s$ and $\mathbf{X}_t$ are marginally independent if they have no neighboring functions in common. We can thus write

$$F(\mathbf{x}) = \prod_{c \in \mathrm{N}(\mathbf{X}_s)} \phi_c(\mathbf{x}_c) \prod_{c \in \mathrm{N}(\mathbf{X}_t)} \phi_c(\mathbf{x}_c) \prod_{c \notin \mathrm{N}(\mathbf{X}_s) \bigcup \mathrm{N}(\mathbf{X}_t)} \phi_c(\mathbf{x}_c). \tag{9}$$

Marginalizing over other variables $\mathbf{X} \setminus \{\mathbf{X}_s, \mathbf{X}_t\}$, we obtain

$$\begin{aligned} F(\mathbf{x}_s, \mathbf{x}_t) &= \sup_{\mathbf{x} \setminus \{\mathbf{x}_s, \mathbf{x}_t\}} F(\mathbf{x}) \\ &= \left(\sup_{\mathbf{x} \setminus \mathbf{x}_s} \prod_{c \in \mathrm{N}(\mathbf{X}_s)} \phi_c(\mathbf{x}_c) \right) \left(\sup_{\mathbf{x} \setminus \mathbf{x}_t} \prod_{c \in \mathrm{N}(\mathbf{X}_t)} \phi_c(\mathbf{x}_c) \right) \\ &\cdot \left(\sup_{\mathbf{x} \setminus \{\mathbf{x}_s, \mathbf{x}_t\}} \prod_{c \notin \mathrm{N}(\mathbf{X}_s) \bigcup \mathrm{N}(\mathbf{X}_t)} \phi_c(\mathbf{x}_c) \right) \\ &= \left(\sup_{\mathbf{x} \setminus \mathbf{x}_s} \prod_{c \in \mathrm{N}(\mathbf{X}_s)} \phi_c(\mathbf{x}_c) \right) \left(\sup_{\mathbf{x} \setminus \mathbf{x}_t} \prod_{c \in \mathrm{N}(\mathbf{X}_t)} \phi_c(\mathbf{x}_c) \right) \\ &= g(\mathbf{x}_s) h(\mathbf{x}_t), \end{aligned}$$

where $\sup_{\mathbf{x}_s} g(\mathbf{x}_s) = \sup_{\mathbf{x}_t} h(\mathbf{x}_t) = 1$ follows from (6), so $F(\mathbf{x}_s) = g(\mathbf{x}_s)$ and $F(\mathbf{x}_t) = h(\mathbf{x}_t)$. Thus, we have $F(\mathbf{x}_s, \mathbf{x}_t) = F(\mathbf{x}_s) F(\mathbf{x}_t)$ and so $\mathbf{X}_s \perp\!\!\!\perp \mathbf{X}_t$. □

An example of the marginal independence property for a 3-variable CDN is shown in Figure 3. Another interesting property of CDNs is that conditioning on variables in the network corresponds to computing derivatives or finite differences of the joint CDF, as we will now show.

Theorem (Conditioning). *Conditioned on a set of variables $\mathbf{X}_s = \mathbf{x}_s$, we have*

$$F(\mathbf{x}_t | \mathbf{x}_s) = \frac{\partial_{\mathbf{x}_s} \Big[F(\mathbf{x}_t, \mathbf{x}_s) \Big]}{\sup_{\mathbf{x}_t} \partial_{\mathbf{x}_s} \Big[F(\mathbf{x}_t, \mathbf{x}_s) \Big]} \propto \partial_{\mathbf{x}_s} \Big[F(\mathbf{x}_t, \mathbf{x}_s) \Big]. \tag{10}$$

Proof. Note that $\partial_{\mathbf{x}_s} \Big[F(\mathbf{x}_t, \mathbf{x}_s) \Big] = F(\mathbf{x}_t | \mathbf{x}_s) P(\mathbf{x}_s) = F(\mathbf{x}_t | \mathbf{x}_s) \sup_{\mathbf{x}_t} \partial_{\mathbf{x}_s} \Big[F(\mathbf{x}_t, \mathbf{x}_s) \Big]$ and so

$$F(\mathbf{x}_t | \mathbf{x}_s) = \frac{\partial_{\mathbf{x}_s} \Big[F(\mathbf{x}_t, \mathbf{x}_s) \Big]}{\sup_{\mathbf{x}_t} \partial_{\mathbf{x}_s} \Big[F(\mathbf{x}_t, \mathbf{x}_s) \Big]}.$$

□

Equation (10) establishes the key operation of conditioning in CDNs: there is one-to-one correspondence between conditioning on a set of variables and computing the derivative/finite difference of the joint CDF with respect to these variables. This is a property we will exploit shortly when we are confronted with the problem of performing inference in a CDN.

Theorem (Conditional Independence). *Let $\mathbf{X}_s, \mathbf{X}_t$ and $\mathbf{X}_u$ be three disjoint sets of variables with no cumulative functions $c \in \mathcal{C}$ connecting any two variables in $\mathbf{X}_s, \mathbf{X}_t$. Then $\mathbf{X}_s \perp\!\!\!\perp \mathbf{X}_t | \mathbf{X}_u$ if every path between any two variables in $\mathbf{X}_s$ and $\mathbf{X}_t$ contains no variables in $\mathbf{X}_s, \mathbf{X}_t, \mathbf{X}_u$.*

Proof. We can marginalize over other variables $\mathbf{X} \setminus \{\mathbf{X}_s, \mathbf{X}_t, \mathbf{X}_u\}$. If this results in a CDN in which there is no path between any two variables in $\mathbf{X}_s$ and $\mathbf{X}_t$ such that all variable nodes in that path are in $\mathbf{X}_u$, then the sets $\{\mathbf{X}_s, \mathbf{X}_a\}, \{\mathbf{X}_t, \mathbf{X}_b\}$ consist of two disjoint subgraphs, with no cumulative functions ϕ_c connecting the two subgraphs and $\mathbf{X}_a, \mathbf{X}_b$ being two disjoint partitions of $\mathbf{X}_u$. Thus we can write

$$F(\mathbf{x}_s, \mathbf{x}_t, \mathbf{x}_u) = \prod_{c \in N(\mathbf{X}_a) \bigcup N(\mathbf{X}_s)} \phi_c(\mathbf{x}_c) \prod_{c \in N(\mathbf{X}_b) \bigcup N(\mathbf{X}_t)} \phi_c(\mathbf{x}_c)$$

$$\begin{aligned} \Rightarrow F(\mathbf{x}_s, \mathbf{x}_t | \mathbf{x}_u) &\propto \partial_{\mathbf{x}_u} \Big[F(\mathbf{x}_s, \mathbf{x}_t, \mathbf{x}_u) \Big] \\ &= \tilde{g}(\mathbf{x}_s, \mathbf{x}_a) \tilde{h}(\mathbf{x}_t, \mathbf{x}_b) \end{aligned}$$

and so $\mathbf{X}_s \perp\!\!\!\perp \mathbf{X}_t | \mathbf{X}_u$. □

In summary, we have $\mathbf{X}_s \perp\!\!\!\perp \mathbf{X}_t | \mathbf{X}_u$ if all paths linking $\mathbf{X}_s$ to $\mathbf{X}_t$ are blocked by unobserved variables not in $\mathbf{X}_s, \mathbf{X}_t, \mathbf{X}_u$. An example of conditional dependence and independence is given in Figure 4.

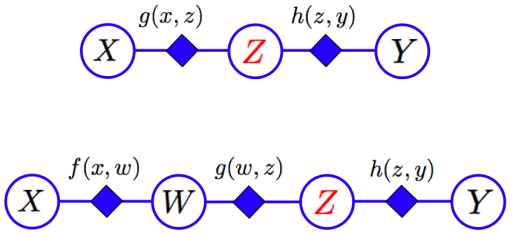


Figure 4: Conditional independence property of CDNs. Two variables X and Y become conditionally dependent given that the variable Z is observed and no unobserved variables block the path from X to Y (*top*). When an unobserved variable W blocks the path, X and Y are conditionally independent given Z (*bottom*).

4 The derivative-sum-product algorithm for inference in cumulative distribution networks

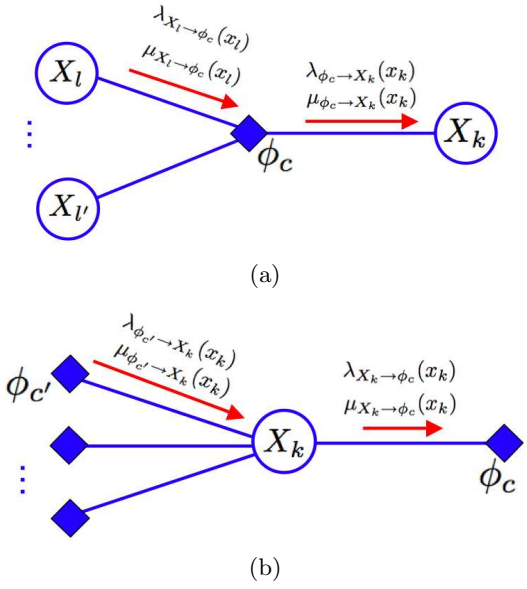


Figure 5: Messages in the derivative-sum-product algorithm: a) Computation of the messages $\mu_{\phi_c \to X_k}(x_k), \lambda_{\phi_c \to X_k}(x_k)$ from a function node ϕ_c to a variable node X_k; b) Computation of the messages $\mu_{X_k \to \phi_c}(x_k), \lambda_{X_k \to \phi_c}(x_k)$ from a variable node X_k to a function node ϕ_c

In this section we derive a *derivative-sum-product* algorithm for computing conditional CDFs efficiently in CDNs. We assume that the CDN has a tree structure to allow for exact inference, with the root variable node of the tree denoted by X_r. We also assume that the functions $\phi_c(\mathbf{x}_c)$ obey all of the necessary and sufficient conditions presented above: in particular we assume that all the functions ϕ_c in the network are differentiable with respect to all of their arguments. In the case in which variables are discrete, we can interchange differentiation with the simpler operation of finite differences. Without loss of generality, we can marginalize over all unobserved variables in the CDN by deleting them from the network and modifying their neighboring cumulative functions appropriately: thus, we assume that the network consists of fully observed variables. In the case we are differentiating (or computing finite differences) with respect to a set of variables which are observed with values $\mathbf{x}_s$, we implicitly assume that the resulting derivative is evaluated at the observed scalar values $\mathbf{x}_s$.

Before proceeding, it is worth asking whether there is a connection between message-passing in a CDN and message-passing in a density model such as a Bayesian network, Markov random field or a factor graph. As noted in Section 1.1, CDNs in general will not have a one-to-one corresponding representation in one of these graphical forms due to the unique conditional independence properties of the CDN. Thus in general, messages passed by the DSP algorithm will therefore be distinct from those exchanged under standard message-passing algorithms such as Pearl's belief propagation [12] or the sum-product algorithm [7] and will not generally correspond to computing the same sufficient statistics.

Our derivation of the DSP algorithm is analogous to that of Pearl's belief propagation algorithm [12]. The intuition here is similar in that we can distribute the differentiation operation to local functions such that each one computes the derivatives with respect to local variables and passes the result to its neighbors in the form of messages $\mu_{\phi_c \to X_k}(x_k)$ from function nodes to variables nodes and $\mu_{X_k \to \phi_c}(x_k)$ from variable nodes to function nodes. Since we assume that the derivatives/finite differences of the functions $\phi_c(\mathbf{x}_c)$ can be computed, we can apply the product rule of differential calculus to expand the messages in the DSP algorithm in terms of derivatives of the functions $\phi_c(\mathbf{x}_c)$ and derivatives of the messages. If we denote $\partial_{x_k}\left[\mu_{\phi_c \to X_k}(x_k)\right] = \lambda_{\phi_c \to X_k}(x_k)$ and $\partial_{x_k}\left[\mu_{X_k \to \phi_c}(x_k)\right] = \lambda_{X_k \to \phi_c}(x_k)$, we can write DSP using two sets of messages μ and λ, as shown in Figure 6.

The joint PDF is then given at the root node X_r by $P(\mathbf{x}) = \lambda_{X_r \to \emptyset}(x_r)$. We need only to pre-compute the derivatives of the ϕ_c's with respect to all their arguments and combine these locally. Note that to compute the joint PDF $P(\mathbf{x})$ from $F(\mathbf{x})$ by brute force for K variables over N functions would require a summa-

- For each leaf variable node X_l and for all functions $c \in \mathrm{N}(X_l)$, propagate $\mu_{X_l \to \phi_c}(x_l) = \lambda_{X_l \to \phi_c}(x_l) = 1$. For each leaf function node $\phi_l(x_l)$, send the messages $\mu_{\phi_l \to X_l}(x_l) = \phi_l(x_l), \lambda_{\phi_l \to X_l}(x_l) = \partial_{x_l}\Big[\phi_l(x_l)\Big]$.

- For each non-leaf variable X_k and neighboring functions $c \in \mathrm{N}(X_k)$,

$$\mu_{X_k \to \phi_c}(x_k) = \prod_{c' \in \mathrm{N}(X_k) \setminus c} \mu_{\phi_{c'} \to X_k}(x_k) \tag{11}$$

$$\lambda_{X_k \to \phi_c}(x_k) = \partial_{x_k}\Big[\mu_{X_k \to \phi_c}(x_k)\Big] = \mu_{X_k \to \phi_c}(x_k) \sum_{c' \in \mathrm{N}(X_k) \setminus c} \frac{\lambda_{\phi_{c'} \to X_k}(x_k)}{\mu_{\phi_{c'} \to X_k}(x_k)} \tag{12}$$

- For each non-leaf function c and neighboring variables $X_k \in \mathrm{N}(\phi_c)$,

$$\mu_{\phi_c \to X_k}(x_k) = \sum_{\substack{s,t | \mathbf{X}_s \bigcup \mathbf{X}_t = \mathbf{X}_c \setminus X_k \\ \mathbf{X}_s \bigcap \mathbf{X}_t = \emptyset}} \partial_{\mathbf{x}_s}\Big[\phi_c(\mathbf{x}_c)\Big] \prod_{j | X_j \in \mathbf{X}_s} \mu_{X_j \to \phi_c}(x_j) \prod_{j | X_j \in \mathbf{X}_t} \lambda_{X_j \to \phi_c}(x_j) \tag{13}$$

$$\begin{aligned} \lambda_{\phi_c \to X_k}(x_k) &= \partial_{x_k}\Big[\mu_{\phi_c \to X_k}(x_k)\Big] \\ &= \sum_{\substack{s,t | \mathbf{X}_s \bigcup \mathbf{X}_t = \mathbf{X}_c \setminus X_k \\ \mathbf{X}_s \bigcap \mathbf{X}_t = \emptyset}} \partial_{\mathbf{x}_s, x_k}\Big[\phi_c(\mathbf{x}_c)\Big] \prod_{j | X_j \in \mathbf{X}_s} \mu_{X_j \to \phi_c}(x_j) \prod_{j | X_j \in \mathbf{X}_t} \lambda_{X_j \to \phi_c}(x_j) \end{aligned} \tag{14}$$

- Repeat the 2^{nd} and 3^{rd} steps above from root to leaf nodes and fuse messages at each variable node to get the conditionals.

Figure 6: The derivative-sum-product message-passing algorithm for computing conditionals in a cumulative distribution network.

tion with a number of product terms of $O(N^K)$. However with derivative-sum-product, we have reduced this to $O(\max\{\max_v(deg(v) - 1), \max_\phi(2^{deg(\phi)-1})\})$, where v represents any variable node in the CDN and ϕ represents any function node.

5 Application: Structured ranking learning using CDNs

In structured ranking learning, we are interested in performing inference on multiple ordinal variables subject to a set of ordinal and stochastic independence/dependence relationships. While it is possible to formulate the problem under the framework of graphical density models, it is quite easy to specify both ordinal and *statistical* independence/dependence relationships between ordinal variables using the CDN framework while allowing for both a flexible model and tractable inference. To demonstrate the application of CDNs to structured ranking learning, we focus on the problem of rank prediction in multiplayer online gaming.

We downloaded the Halo 2 Beta Dataset[2] consisting of player performance scores for 4 game types ("LargeTeam","SmallTeam","HeadToHead' and "FreeForAll") over a total of 6465 players and thousands of games. For each game, an outcome is defined by the performances of teams once the game is over, where team performances are determined by adding players performances together. Recently, [4] presented the TrueSkill algorithm for skill rating, whereby each player is assigned a distribution over skill levels which are inferred from individual player performances over multiple games using expectation propagation [10].

Our model is designed according to two principles: firstly, that the relationship between player scores and game outcomes is stochastic, and secondly, that game outcomes are determined by teams competing with one another on the basis of team performances. To address the first point, we will require a set of CDN functions which link player scores to team performance. Interpreting team performances as outputs, we will make use of the cumulative model [9], which relates a function $f(\mathbf{x})$ of the input variables $\mathbf{x}$ to the output ordinal variable $y \in \{r_1, \cdots, r_K\}$ so that $Pr\{y = r_i\} = Pr\{\theta(r_{i-1}) < f(\mathbf{x}) + \epsilon \leq \theta(r_i)\} = F_\epsilon(\theta(r_i) - f(\mathbf{x})) - F_\epsilon(\theta(r_{i-1}) - f(\mathbf{x}))$, where ϵ is additive noise and we define $\theta(r_0) = -\infty, \theta(r_K) = \infty$. Here, we will choose $\mathbf{x}$ as the set of player scores for a given team and so the team performance is given by $f(\mathbf{x}) = \mathbf{1}^T\mathbf{x}$. So if for any given game, there are N teams, then each team is assigned a function g_n such

[2]Credits for the use of the Halo 2 Beta Dataset are given to Microsoft Research Ltd. and Bungie.

that

$$g_n(\mathbf{x}_n, r_n) = \int_{-\infty}^{\mathbf{x}_n} F\big(\theta(r_n); \mathbf{1}^T\mathbf{u}, \beta^2\big) P\big(\mathbf{u}\big)\, d\mathbf{u} \quad (15)$$

where $F\big(\theta(r_n); \mathbf{1}^T\mathbf{u}, \beta^2\big)$ is a cumulative model relating player scores to team performance, $\theta(r_n)$ are "cutpoints" which define contiguous intervals in which the team performance is assigned as $y_n = r_i$ and $P(\mathbf{u})$ is a density over the vector of player scores $\mathbf{u}$. We will model these functions as

$$F\big(\theta(r_n); \mathbf{1}^T\mathbf{u}, \beta^2\big) = \Phi\big(\theta(r_n); \mathbf{1}^T\mathbf{u}, \beta^2\big), \quad (16)$$

$$P(\mathbf{u}) = \mathcal{N}(\mathbf{u}; \mu\mathbf{1}, \sigma^2\mathbf{I}), \quad (17)$$

where $\Phi(\cdot; m, s^2)$ is the cumulative distribution function for a univariate Gaussian with mean m and variance s^2, and $\mathcal{N}(\cdot; \boldsymbol{m}, \mathbf{S})$ is a multivariate Gaussian density.

To address the fact that teams compete for higher rank, we model ordinal relationships between team performance so that for two teams with performances R_X, R_Y, $R_X \preceq R_Y$ if $F_{R_X}(\alpha) \geq F_{R_Y}(\alpha)\, \forall \alpha$, where $F_{R_X}(\cdot), F_{R_Y}(\cdot)$ are the marginal CDFs of R_X, R_Y. Given N ranked teams, we can thus define $N-1$ functions $h_{n,n+1}$ so that

$$h_{n,n+1}(r_n, r_{n+1}) = \Phi\left(\begin{bmatrix} r_n \\ r_{n+1} \end{bmatrix}; \begin{bmatrix} \tilde{r}_n \\ \tilde{r}_{n+1} \end{bmatrix}, \mathbf{\Sigma} \right)$$

where

$$\mathbf{\Sigma} = \begin{bmatrix} \beta^2 & \rho\beta^2 \\ \rho\beta^2 & \beta^2 \end{bmatrix}$$

and $\tilde{r}_n \leq \tilde{r}_{n+1}$ so as to enforce $R_n \preceq R_{n+1}$ in the overall model. The CDN corresponding to our model is shown in Figure 7(a): one can verify for any given game that indeed the relationship $R_1 \preceq R_2 \preceq \cdots \preceq R_N$ is enforced by marginalizing over player scores. Our model includes a function $s_k(x_k)$ for each player k which represents that player's skill. The goal is then to learn the player skills: we used an online learning method for updating the functions $s_k(x_k)$, whereby for each game we perform message-passing followed by the update $s_k(x_k) \leftarrow s_k(x_k)\mu_{g_n \to X_k}(x_k)$ for each player. Before each game, we predict the outcome using the player skills learned thus far via $x_k^* = \arg\max_{x_k} \partial_{x_k}\Big[s_k(x_k)\Big] = \arg\max_{x_k} \lambda_{X_k \to s_k}(x_k)$.

Since all of the CDN functions are themselves multivariate Gaussian CDFs, the derivatives $\partial_{\mathbf{x}_s}\Big[\phi_c(\mathbf{x}_c)\Big]$ in the corresponding message-passing algorithm can be easily evaluated w.r.t. variables $\mathbf{X}_s$ as

$$\partial_{\mathbf{x}_s}\Big[\Phi\big(\mathbf{x}; \boldsymbol{\mu}, \mathbf{\Sigma}\big)\Big] = \mathcal{N}\Big(\mathbf{x}_s; \boldsymbol{\mu}_s, \mathbf{\Sigma}_s\Big)\Phi\Big(\mathbf{x}_t; \tilde{\boldsymbol{\mu}}_t, \tilde{\mathbf{\Sigma}}_t\Big)$$

where

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_s \\ \mathbf{x}_t \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_s \\ \boldsymbol{\mu}_t \end{bmatrix}, \quad \mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_s & \mathbf{\Sigma}_{s,t} \\ \mathbf{\Sigma}_{s,t}^T & \mathbf{\Sigma}_t \end{bmatrix},$$

$$\tilde{\boldsymbol{\mu}}_t = \boldsymbol{\mu}_t + \mathbf{\Sigma}_{s,t}^T\mathbf{\Sigma}_s^{-1}(\mathbf{x}_s - \boldsymbol{\mu}_s),$$
$$\tilde{\mathbf{\Sigma}}_t = \mathbf{\Sigma}_t - \mathbf{\Sigma}_{s,t}^T\mathbf{\Sigma}_s^{-1}\mathbf{\Sigma}_{s,t}.$$

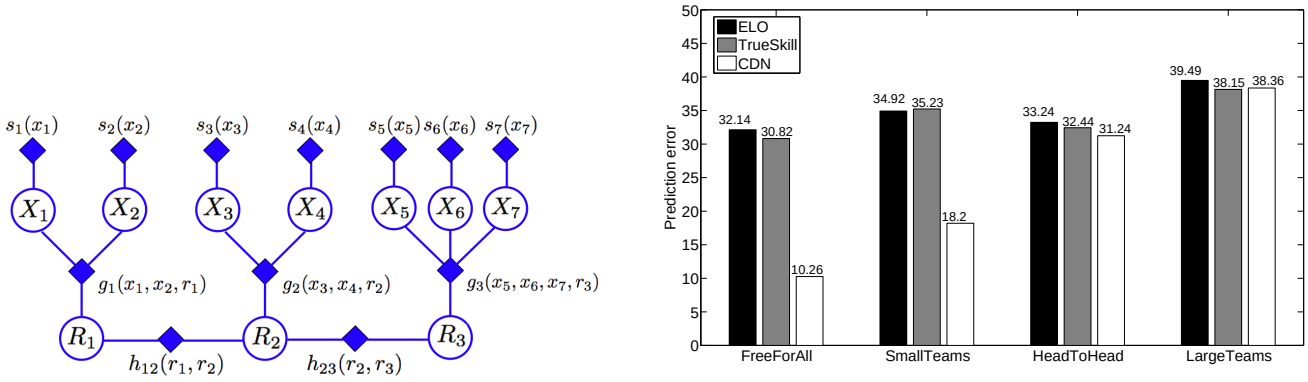
To learn the model, we set the cutpoints $\theta(r_n)$ in the above model using ordinal regression of team performances on team scores, with μ set to the minimum player score. A plot showing the total prediction error rate of our method is shown in Figure 7(b), along with the error rates reported by [4] for TrueSkill and ELO [2], which is a statistical rating system used in chess.

6 Discussion and future work

We proposed a novel class of probabilistic graphical models, the cumulative distribution network (CDN). We demonstrated that the CDN has distinct statistical independence properties from Bayesian networks, Markov random fields and factors graphs. This also differs from convolutional factor graphs [8], which replace products of functions with convolutions whilst retaining the same conditional independence and dependence structure, albeit in the Fourier domain.

The task of computing marginals in CDNs is achievable in constant time. We proposed the derivative-sum-product (DSP) algorithm for computing these conditional distributions and demonstrated it to be a distinct from standard message-passing algorithms such as Pearl's belief propagation or the sum-product algorithm. We focused here on the application of CDNs to the problem of ranking learning in a multiplayer gaming setting. A list of other potential applications (which is by no means exhaustive) would include ranking webpages, collaborative filtering, inference of censured data [3] and generalized hypothesis testing in which one could both model statistical dependencies between multiple hypotheses.

While we have presented the DSP algorithm for computing conditionals given a set of cumulative functions, we have not addressed here the issue of how to learn these cumulative functions from data. An obvious method would be to run derivative-sum-product to obtain the joint PDF and then maximize this with respect to model parameters, but this approach has yet to be worked out in detail. Another issue we have not addressed is how to perform inference in graphs with cycles: an interesting future direction would be to investigate exact or approximate methods for doing so and connections to methods in the literature [5, 11] for doing this in traditional graphical models.



(a) (b)

Figure 7: a) An example CDN corresponding to a multiplayer game with 3 teams of 2, 2 and 3 players with stochastic team rankings $R_1 \preceq R_2 \preceq R_3$. The first layer of functions $s_k(x_k)$ correspond to distributions over player performances (or *skills*). The functions g_n, h_n relate team ranks to to one another as well as player performances; b) Prediction error (computed as the fraction of team predicted incorrectly before each game) achieved by our method, compared to ELO [2] and TrueSkill error [4].

7 Acknowledgements

We would like to thank anonymous reviewers, Radford Neal and Frank Kschischang for helpful comments. J.C.H. was supported by a National Science and Engineering Research Council postgraduate scholarship. This research was supported by an NSERC Discovery Grant.

References

[1] Buntine, W. (1995) Chain graphs for learning. *In Proceedings of Conference on Uncertainty in Artificial Intelligence 11 (UAI 1995)*, 46-54.

[2] Elo, A.E. (1978) The rating of chess players: Past and present. *Arco Publishing, New York.*

[3] Gelman, A., Carlin, J.B., Stern, H.S. and Rubin, D.B. (2004) Bayesian data analysis, Second edition. *Texts in Statistical Science, Chapman & Hall.*

[4] Herbrich, R., Minka, T.P. and Graepel, T. (2007) TrueSkill^{TM}: A Bayesian skill rating system. *In Proceedings of Neural Information Processing Systems 2007 (NIPS 2007)*, 569-576.

[5] Jordan, M.I., Ghahramani, Z., Jaakkola, T.S. and Saul, L.K. (1999) An introduction to variational methods for graphical models. *Machine Learning* **37(2)**, 183-223.

[6] Kinderman, R. and Snell, J. L. (1980) Markov random fields and their applications. *American Mathematical Society.*

[7] Kschischang, F.R., Frey, B.J. and Loeliger, H.-A. (2001) Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory* **47(2)**, 498-519.

[8] Mao, Y., Kschischang, F.R. and Frey, B.J. (2004) Convolutional factor graphs as probabilistic models. *In Proceedings of Conference on Uncertainty in Artificial Intelligence 20 (UAI 2004)*, 374-381.

[9] McCullagh, P. (1980) Regression models for ordinal data. *J. Royal Statistical Society, Series B (Methodological)* **42(2)**, 109-142.

[10] Minka, T.P. (2001) Expectation propagation for approximate Bayesian inference. *In Proceedings of Conference on Uncertainty in Artificial Intelligence 17 (UAI 2001)*, 362-369.

[11] Neal, R. M. (1993) Probabilistic Inference Using Markov Chain Monte Carlo Methods, *Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto*, 144 pages.

[12] Pearl, J. (1988) Probabilistic reasoning in intelligent systems. *Morgan Kaufmann.*

Toward Experiential Utility Elicitation for Interface Customization

Bowen Hui
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada

Craig Boutilier
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada

Abstract

User preferences for automated assistance often vary widely, depending on the situation, and quality or presentation of help. Developing effective models to learn individual preferences online requires domain models that associate observations of user behavior with their utility functions, which in turn can be constructed using utility elicitation techniques. However, most elicitation methods ask for users' *predicted* utilities based on hypothetical scenarios rather than more realistic *experienced* utilities. This is especially true in interface customization, where users are asked to assess novel interface designs. We propose experiential utility elicitation methods for customization and compare these to predictive methods. As experienced utilities have been argued to better reflect true preferences in behavioral decision making, the purpose here is to investigate accurate and efficient procedures that are suitable for software domains. Unlike conventional elicitation, our results indicate that an experiential approach helps people understand stochastic outcomes, as well as better appreciate the sequential utility of intelligent assistance.

1 Introduction

Intelligent software customization has become increasingly important as users are faced with larger, more complex applications. For a variety of reasons, software must be tailored to specific individuals and circumstances [21]. For example, adaptive interfaces are critical as different users may: require different functionality from multi-purpose software [5]; prefer different modes of interaction; use software on a variety of hardware devices [12]; or, due to expanding software complexity, require online and automated help to identify and master different software functions [16]. In the latter case, a system should ideally adapt the help it provides and the decision to interrupt a user [15] to account for specific user preferences.

In this paper, we focus on *interface customization* where the attributes of interface widgets (e.g., location, transparency, and functionality) are automatically tailored to the needs of specific users. In particular, we are interested in intelligent systems that learn to predict user goals over time based on observed user behavior, and suggest ways (e.g., through interface customization) to help the user complete the desired goal. Considerable work has been devoted to the prediction of user needs and goals (e.g., [16; 1; 28; 3] among others), much of it is focused on developing probabilistic models of user goals. Less emphasis has been placed on assessing user preferences for software interaction and customization (for exceptions, see [11; 17]). However, accounting for user preferences is critical to good interface customization. For instance, consider automated word completion [10]. Some users may prefer single-word suggestions, while others may prefer several different suggestions. Similarly, some users may be satisfied with "partial help" (e.g., a partially correct word that saves a few keystrokes) while others may wish to use only completely correct completions. These preferences, and more importantly, a user's *strength of preference*, are needed to make suggestion decisions: preferences must be weighed against the probability of specific user goals.

In this paper, we evaluate the effectiveness of a variety of preference elicitation techniques for interface customization. While elicitation may not be used directly (i.e., online) during application use, most adaptive systems will make indirect assessments of user preferences (e.g., [32; 30; 8; 19]). However, even indirect assessment methods require some knowledge of the range of possible user preferences and how those are (perhaps stochastically) related to observable behavior. In these cases, offline preference elicitation for different customizations can provide valuable data for the design of an online system.

Most existing literature on preference elicitation in AI, as well as the majority of that in behavioral decision theory, assumes that people "know" their preferences *a pri-*

ori. However, often users have not encountered, nor even considered, the hypothetical situations typically posed in the elicitation process. This is especially true of software customization, since the situations involve novel interfaces. Under these circumstances, people may report their *predicted utilities* by conceptualizing the posed scenarios and forecasting their own preferences. However, what people "think they like" can systematically differ from what they "actually like" [23]. For example, someone who has not actually engaged in a system that offers a range of partial word completion suggestions may predict that they dislike the interface in a particular circumstance, but in fact like it when they experience it (or vice versa).

For this reason, we propose a novel *experiential elicitation* approach for interface customization. Our elicitation "queries" allow users to assess *experienced utilities* [25] by providing simple, hands-on tasks and system suggestions or customizations, drawn from a particular distribution. We adapt standard elicitation approaches to incorporate such *experiential queries*. Our approach also overcomes some of the difficulties with well-established procedures (e.g., standard gambles) that involve probabilities over a distribution of outcomes. We explore this new approach in the context of a specific customization task—the suggestion of highlighting options in PowerPoint — and show that experiential elicitation offers a more accurate means of assessing quantitative tradeoffs in preferences. Unfortunately, one drawback of experiential queries is the time they take. To counteract this, we also propose two hybrid models that attempt to assess experienced utilities somewhat more (time) efficiently. Our results show that one hybrid procedure provides a good approximation to the experiential approach in a much more effective manner.

In Section 2, we outline the basic customization domain, and describe the underlying decision-theoretic model used to provide assistance to users. We describe essential background on preference elicitation in Section 3. Our approach to experiential elicitation for interface customization is presented in Section 4, as is our empirical evaluation. In Section 5 we consider two hybrid approaches that accelerate the experiential process, *primed* and *primed+* elicitation, and evaluate their effectiveness. The key benefit in our experiential approach is that it enables users to better interpret queries and assess the sequential utility of an interface via hands-on experience. As a result, our approach can provide a more accurate picture of the user's preferences.

2 The Customization Domain

We are interested in developing intelligent systems that perform online interface customization based on user preferences and user needs. We focus on a concrete form of customization as an example to illustrate our approach, but note that the general principles apply more broadly. We use this domain and application to motivate the need for preference elicitation for interface customization and to highlight the difficulties with standard elicitation methods. Our main contributions are discussed in Sections 4 and 5, but will apply more broadly than the task considered here.

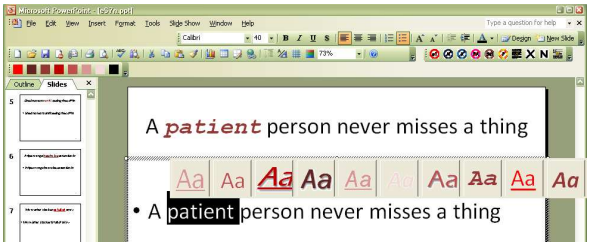


Figure 1: Icon suggestions to help the user in a highlighting task implemented as part of PowerPoint 2003.

2.1 The Highlighting Task

Consider a user authoring slides in PowerPoint who wishes to highlight important phrases and new terminology by applying a particular font stylization. For aesthetic reasons, the user tends to choose from just a few highlighting styles that are consistent throughout a presentation, and possibly across multiple presentations. This consistency provides the opportunity for an intelligent system to observe and learn user-specific patterns, so that it can offer useful suggestions in the future. Figure 1 shows an example toolbar with 10 icons, each suggesting a particular set of font characteristics that can be applied to highlight the selected phrase "patient." Many repetitive tasks in PowerPoint and other software can benefit from automated suggestions designed to minimize user effort.

We assume that in a highlighting task, the user has a certain style in mind, which involves changing some number of font *features* to make it stand out from the rest of the text. We define the *complexity* of a highlighting style as the number of font feature values required to create it. In other words, this is the number of events that the user must execute to make a phrase different from neighboring text or some baseline default font style. For example, relative to "plain" black text, a bold, italicized font has complexity 2.

To assist the user with the highlighting task, our system can suggest a toolbar with a number of icons, each offering some combination of font features shown by the characters "Aa" (as illustrated in Figure 1). The user may: select one of these icons to apply the associated font style; complete the highlighting task purely manually using the mouse (e.g., click on the Bold icon) or shortcut keys (e.g., press Ctrl-B); or accept a suggestion, but further refine it manually by applying (or undoing) additional font characteristics. If the toolbar is ignored (e.g., the user continues typing), it disappears after a short time.

2.2 Predictive Model & Assistance Decisions

Intuitively, the value of highlighting suggestion depends on the amount of savings it offers relative to manual task completion. There are also costs to suggestions: interruption, processing costs, mode switching, etc. We discuss the relative value and costs of a set of toolbar suggestions below. Notice, however, that the value of a suggestion cannot be known with certainty: the system can only make a stochastic prediction about the user's true intended goal. These predictions must be weighed against the overall costs and benefits of making (or not making) a suggestion.

While our aim is not to discuss predictive models, we give a sketch of our system model in order to place our elicitation results in the appropriate context. We focus specifically on *highlighting goals* in which the user desires a certain style (combination of font attributes). The system observes past user events and learns user-specific styles, which are stored in a goal library. For each goal, the system creates a probabilistic event model (a stochastic automaton), and at runtime maintains a distribution over goals given the stream of observed user events. Suggestions are made (or not) decision-theoretically using a partially observable Markov decision process (POMDP) [2] to tradeoff goal probabilities with the costs and benefits of various suggestions [19] given specific user preferences. We elaborate on the cost model in Section 2.3, as well as how user preferences can be incorporated into a POMDP in Section 3.

2.3 The Value and Costs of Suggestions

By selecting one of the suggested icons, the user saves the effort of manually completing the task herself (or some part of the task). *Savings* is an objective measure of help quality, reflecting the number of steps/actions a user avoids by accepting a suggestion. Common examples that provide savings are auto-completion and the Office 2007 mini-toolbar. Research in human-computer interaction (HCI) suggests that the manual effort of interaction (e.g., moving the mouse, typing, mode switching) is a function of the user's actions, the number of such actions, and the modes used to execute them [6]. The quality of help actions in adaptive systems can be defined similarly, capturing the difference between manual effort required by the user with and without help. We define the quality $Q(i|g)$ of a suggestion icon i (given a goal g) to be this difference. Since we expect users to pick the best icon available, we define the quality of toolbar t to be $Q(t|g) = \max_{i \in t} Q(i|g)$, the maximum effort saved by any icon in the toolbar.[1]

The subjective value of help may vary depending on certain user features, such as neediness and distractibility [16; 19], frustration/distress [26; 9; 19], and independence [19]. In other words, quality of the suggestion alone may not predict system utility. For example, someone who currently needs help with a difficult task may benefit greatly from partial suggestions that help the user identify the next steps, while someone who is highly independent may not even accept (or find value in) suggestions of perfect quality. In other words, the *perceived utility* of automated help is a function of certain user characteristics, such as how much help the user needs or how independent the user is. In a probabilistic model, these characteristics are hidden user variables that need to be estimated online based on sequences of observed user behavior. In our initial design, we focus on neediness only. Examples of observable characteristics that a system can use to infer the user's neediness level include the user being stuck (i.e., pausing during task activity) or looking for help (i.e., browsing without selection).[2] These observations often arise in difficult tasks. Therefore, we define the user's level of neediness $N(g)$ as a function of how difficult the goal g is to the user. (We explain how we simulate user neediness in a controlled experiment in Section 4.2.)

Apart from potential savings, toolbar suggestions also have associated costs. In general, *information processing* refers to the user scanning and evaluating a set of items of similar nature. Common interface examples that require information processing are menus and toolbars. Research in HCI suggests that the time it takes a user to process a set of items in an interface is a function of the number of items and the search strategy used based on the expertise level of the user [13; 22; 14]. In adaptive toolbars, the icons are always changing, so users cannot develop a search strategy to minimize processing time. To model processing time, we focus only on the number of items in an adaptive toolbar. We define length $L(t)$ to be the number of icons in t.

Depending on the specific system action, costs other than information processing may be relevant in determining system utility. We refer interested readers to [20] for a detailed discussion of interaction cost models.

3 Preference Elicitation

In order for the system to choose a good toolbar to help the user, it needs a model of the user's preferences for possible suggestions. The value of a suggestion t depends on the user's utility function with respect to user neediness N, toolbar length L, and suggestion quality Q. Let O be the set of possible outcomes over the values of these three attributes. We use notation such as $n1, l5, q4$ to represent the outcome with the user's neediness level at 1 (high) and the toolbar showing five icons with toolbar quality 4. The need for a utility function (rather than qualitative preferences) should be apparent given the stochastic nature of goal estimation.

[1] Of course, since the goal is only known stochastically, expected quality must be computed relative to the system's beliefs about the user's current highlighting goal.

[2] Bayesian models exist for learning neediness [16; 19].

A user's preferences for particular outcomes, including their strength of preference, can be represented by a *utility function*, $u : O \rightarrow \mathbb{R}$, where $u(o_i) > u(o_j)$ iff o_i is preferred to o_j, and $u(o_i) = u(o_j)$ iff the user is indifferent between o_i and o_j. For convenience, we normalize utilities to the interval $[0, 1]$, defining $o^\top$ to be the best outcome with $u(o^\top) = 1$ and $o_\perp$ to be the worst outcome with $u(o_\perp) = 0$. A utility function can be viewed as reflecting qualitative preferences over *lotteries* (distributions over outcomes) [27], with one lottery preferred to another iff its expected utility is greater. Let $SG(p) = \langle p, o^\top; 1\text{-}p, o_\perp \rangle$ denote a *standard gamble*, a specific (parameterized) lottery where $o^\top$ is realized with probability p and $o_\perp$ is realized with probability $1 - p$. The expected utility of this lottery is p.

The *standard gamble query* (SGQ) for outcome o_i asks the user to state the probability p for which she would be indifferent between $SG(p)$ and outcome o_i [27]. This type of query is extremely informative as it asks the user to assess a precise tradeoff involving o_i, and indeed fixes $u(o_i)$ on the normalized scale. However, this makes such queries practically impossible to answer with confidence. More cognitively plausible are *bound queries*. The bound query $B(o_i, p)$ asks the user whether she would prefer $SG(p)$ to o_i. A positive response places an upper bound of p on $u(o_i)$, while a negative response places a similar lower bound. Their yes/no nature makes bound queries easier for people to answer. In principle, bound queries can be used to incrementally elicit utility functions to any required degree of precision by incrementally refining the bounds on utility outcomes, at each stage giving rise to a more refined set of *feasible* utility functions (those consistent with the bounds). In practice, as the feasible regions for each parameter become small, the queries will generally become harder to answer with confidence.

In practice, a lottery is presented as textual (or verbal) description of two outcomes and their probabilities, possibly accompanied by visual aids representing the outcomes. We refer to this delivery of bound query as a *conceptual query*, since people are asked to think about the two alternatives before making a decision. During conceptual elicitation in our domain, each query involves asking the user to think about the two options by imagining the use of two separate systems to complete the highlighting task. $SG(p)$ corresponds to using an adaptive system repeatedly, with percentage p of the instances involving the best interface $o^\top$ and the rest $o_\perp$. Option o_i corresponds to using a static system repeatedly.

Though theoretically appealing, people often have difficulty assessing the probability parameter in SGQs [27]. Similarly, bound queries with precise probabilities can be hard to conceptualize in this domain, with people having difficulty comparing interface "lotteries" with a fixed outcome. Finally, a user's assessment of what they like in hypothetical settings can differ systematically from what they actually like [25; 23; 24; 18]. For these reasons, we investigate alternative elicitation mechanisms that overcome these difficulties.

Preference elicitation in interface customization has typically adopted a qualitative approach to assessing user preferences that learns preference rankings without learning the strength of those preferences [29; 31; 11] (although Horvitz et al. [17] uses "willingness to pay" to quantify the cost of interruption). However, given the stochastic nature of goal estimation, we require estimates of utility as discussed above. To do this within our POMDP, we require some means of associating observed user behavior with utility functions. While direct online elicitation is not (completely) feasible, offline elicitation can be used to develop the required models. For instance, utility functions elicited offline can be clustered into a small set of *user types* [7], which the POMDP assesses online. Online assessment of user types may be done passively [19] or explicitly through active elicitation [4]. In this context, we propose to use an experiential elicitation procedure to carry out offline elicitation experiments with real users, which we describe in the next section.

4 Experiential Elicitation

To facilitate interpretation of bound queries, we develop an *experiential* version of bound queries which allows the user to "experience" both query options (including the stochastic one) before stating a preference. The user is asked to complete k simple tasks using the system in each of two ways: the $SG(p)$ option shows $o^\top$ in fraction p of the k tasks and $o_\perp$ in the remaining tasks (in random order). In this way, the user "experiences" a stochastic mixture of interfaces. The deterministic option also requires k task completions, but all using the same interface o_i. After each option, the user is asked to reflect on what she liked and disliked about the experience — which could be a function of the effort required by the tasks, toolbar "processing" cost, the satisfaction in having toolbar help available, or the ease of interaction — and to indicate her preference. The response provides a bound at p for o_i.

4.1 Experiential Elicitation for Interfaces

We elicit $U(N, L, Q)$ from users using the interface shown in Figure 1. We use $k = 10$ tasks for each alternative in the query, so each query involves the user completing 20 highlighting tasks. To reduce the cognitive burden and experiment time, we elicit only $U(N, L, Q)$ for the values $Q \in \{0, 2, 4\}$, $L \in \{1, 5, 10\}$, and $N \in \{0, 1\}$. This discretization yields a range of 18 outcomes. Similarly, we discretize query probabilities: $[0, 0.1, 0.2, ..., 1.0]$. We define $o^\top = n0, l1, q4$, since the user is at a low level of neediness and receives the best help possible, and

$o_\perp = n1, l10, q0$, since the user is at a high level of neediness and receives the worst help possible.

When the system suggests a toolbar, the user can select one icon or ignore it altogether. If the wrong font style is chosen, or the suggestion is ignored, the user must carry out the highlighting task manually. Each task requires the user to carry out several separate actions, and applying an incorrect style requires additional fixes.

4.2 Conceptual vs. Experiential Elicitation

We compared experiential and conceptual queries experimentally to investigate their impact on elicitation based on several criteria: the efficiency (duration) of the procedure; the cognitive demand imposed on users; the interpretability or understandability of queries by the user; and the quality of the elicited responses. In particular, we compare the elicited utilities quantitatively under these two conditions, and test whether mean utilities elicited under conceptual elicitation are the same as those under experiential elicitation (our null hypothesis, H_0). We also examine the general structure of $U(N, L, Q)$ to see if users perceive value in such simple help, and if any trends across the user population exist in the underlying preferences.

In addition to verbal descriptions, we used screenshots to represent each outcome in the Conceptual condition. Thirteen people participated in the Conceptual condition and 8 people in the Experiential condition.

To control for the user's highlighting goal, we define a *target font style* in each task. As illustrated in Figure 1, each PowerPoint slide has two sentences with the same words, where the top sentence indicates the phrase highlighted with the target font style. The user must match the first sentence by changing the font style of the appropriate words in the second. The vocabulary of target font styles is defined by 7 features—5 of which are binary (bold, underline, italics, shadow, size increment) and 2 of which are multi-valued (8 colors and 10 font families). All target font styles in the experiment have complexity 4. Therefore, if the system's suggestion is perfect, it can save the user from manually executing (sequences of) 4 separate events.

Recall that neediness is a hidden user variable, and how much help a user needs is a function of how difficult the user's goal is to accomplish. In the experiment, we simulate two neediness settings by controlling the task environment, and thus, making the user goal more difficult to achieve. To simulate a needy user ($n1$), we make the task more difficult by restricting the set of colors to 7 shades of red and the set of font families to 4 similar fonts. In this way, the system's feature vocabulary, the target font styles in the highlighting tasks, and the icons in the toolbar are restricted to similar colors and fonts. The interface in Figure 1 shows an example of this neediness setting. The full default feature vocabulary defines the interface environment for a user who is not needy ($n0$).

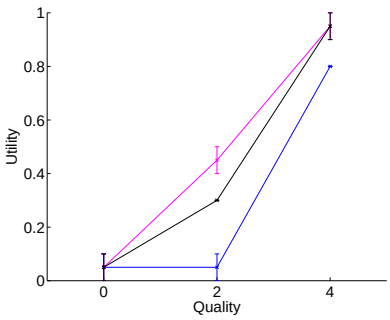


Figure 2: Partial utility functions for $n0, l10$ as a function of Q elicited from 3 users in the Experiential condition. Lines are drawn at the midpoints of the resulting bounds.

During the elicitation, we posed bound queries and incrementally refined the bounds by choosing p to be the midpoint of the set of feasible utility functions until all the outcomes have feasible regions with range ≤ 0.1. Figure 2 illustrates sample results from a specific elicitation run, showing a partial utility function (for fixed values of N and L) for three users. The elicited bounds are drawn as error bars (note that discretization prevents us from pinning down the utility function precisely). The "blue" user (lower line), for instance, has $0 \leq u(n0, l10, q0) \leq 0.1$. The bound at $q4$ for the blue user is tight because she is indifferent between $SG(.8)$ and $o_i = n0, l10, q4$.

Methodological Comparison On average, each experiment took 30 minutes in the Conceptual condition and 2 hours (divided into two sessions) in the Experiential condition. Experiments in the Experiential condition are much longer because experiential queries require users to carry out tasks, while the conceptual queries only require users to think about scenarios.[3]

Since experiential queries require a series of task completions, users became tired early on and found it necessary to take breaks in order to not be confused with the various options and associated experiences. Users in the Conceptual condition did not seem tired during the procedure, but they were at times inconsistent with previous responses.

Although experiential queries took longer, they provided hands-on experience and therefore required little verbal explanation. In contrast, conceptual queries were often diffi-

[3]Our aim is to engage in *complete* utility elicitation (to the prescribed accuracy) to develop models of (classes of) user preferences that can be used in online assessment (e.g., within a POMDP). For this reason, we do not consider means to "intelligently" assess only the *relevant* preferences, using, say, value of information with respect to a specific task, something which is vital in online assessment. Of course, if general domain constraints are known to render various outcomes impossible, we could prune the elicitation task somewhat.

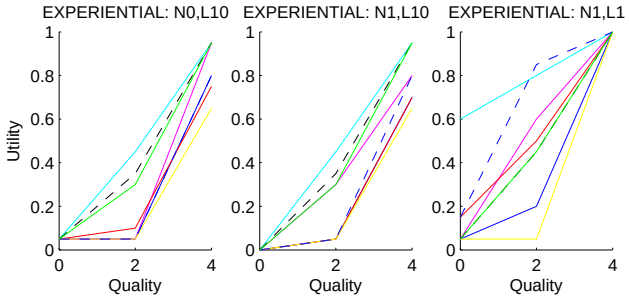


Figure 3: Partial utility functions for $n0, l10$, $n1, l10$, and $n1, l1$ as a function of Q elicited from all 8 users in the Experiential condition. Each line connects the midpoints of the outcome's elicited bounds for each user.

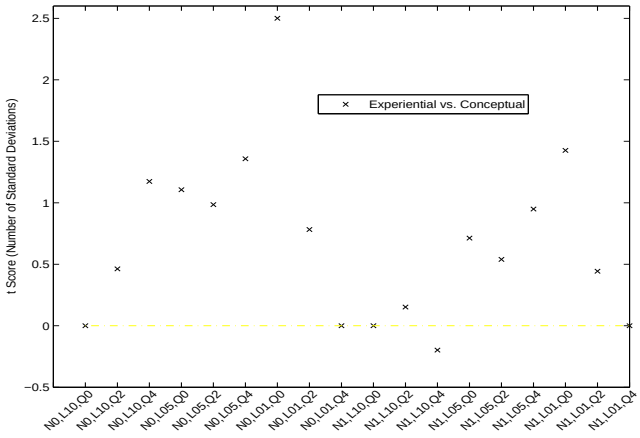


Figure 4: Resulting t scores for each elicited outcome.

cult to explain, because they require users to first understand the aspects of the interface (e.g., amount of effort needed in manually completing the tasks, controlled quality of help in the suggestions), then compare the costs and benefits of a mixture of interfaces with a definitive interface, and finally, imagine using the respective interfaces in a repeated scenario.

Structural Comparison Independently of the elicitation method, we are interested in the perceived value of this adaptive form of customization. The utility functions across the 21 users varied widely—some are convex, some are concave, some are linear, some are "flat" when quality is not perfect, and some are "flat" when length is not one. Some examples (using midpoints of feasible regions) are shown in Figure 3. Clearly, user preferences vary widely, even for such simple highlighting help with three customization attributes.

In general, the utility functions are monotonically non-decreasing in Q when N and L are fixed, and monotonically non-increasing in L when N and Q are fixed. In particular, when help quality is high ($q4$), utility decreases slightly as L increases. This is expected as users perceive higher processing costs with more icons. We also see that *partial help* ($q2$) with L at $l1$ is qualitatively different than $l5$ or $l10$, because more icons decrease the chance of the user identifying the single icon that provides partial help. When help quality is low ($q0$), some users prefer to see one bad suggestion ($l1$) than many bad suggestions ($l10$). There are no general trends in utility given neediness; some users showed clear differences between the needy ($n1$) and not needy ($n0$) scenarios, while others viewed them the same. From this, we see that users perceive value in automated help, even in simple tasks such as highlighting. Of course, more data is needed to draw definitive conclusions about possible parametric forms for utilities that could further simplify online assessment.

Quantitative Comparison Using Hotelling's T^2 statistic,[4] we found that the mean utilities in the two conditions are significantly different, $p < 0.01$. Thus, we reject H_0. To provide a more detailed comparison, we carried out a two-tailed t-test with independent means for each outcome. The resulting t scores[5] are plotted in Figure 4. Only one outcome, $n0, l1, q0$, is *individually* significantly different between the two conditions (with 19 degrees of freedom, a t score of at least ± 2.093 is needed for significance at $p < .05$).

More interestingly, Figure 4 shows that the mean t score tends to be higher for most outcomes in the Experiential condition, including outcomes that provide partial quality help ($q2$) and incorrect suggestions ($l5, q0$ and $l1, q0$). This indicates that users in the Experiential condition perceive greater value in adaptive help than users in the Conceptual condition. The Experiential condition requires users to carry out 20 tasks for each query, while the Conceptual condition only asked users to "think about" the tasks. With conceptual queries, we believe that participants are less likely to truly perceive the value of automated help in repeated scenarios, and thus, underestimate the utility of these outcomes. We believe that experientially assessed utilities more accurately reflect the users' true preferences; however, our experimental set up does not allow us to draw such a definitive conclusion.

5 Ways to Improve the Experiential Elicitation Procedure

Although it seems that experiential queries enable users to report more realistic preferences, the procedure is time consuming (even with simple utility functions over 18 out-

[4] The T^2 distribution is a multivariate analog of the Student's t-distribution. We used the Moore-Penrose pseudoinverse in computing T^2.

[5] A t score is a measure of how far apart the two sample means are on a distribution of differences between means.

comes). Though our intent is to examine methods for offline elicitation to support models for online adaptation—thus we do not face the demands on online customization here—even for offline model development this procedure may be too demanding. We develop two more efficient elicitation procedures, based on the findings in Section 4.2. Following the same experimental set up, we introduce two procedures, *primed* and *primed+*, that attempt to elicit experienced utility more effectively. The Primed condition uses a training session to familiarize users with the interface and the attributes N, L, Q; but the elicitation procedure itself still relies on conceptual queries only. The Primed+ condition uses this training session plus 5 experiential queries at the start of the elicitation. The remaining elicitation is done using conceptual queries only.

Similar to the previous experiment, we want to test whether the mean utilities elicited under the Conceptual condition are the same as those elicited under the Primed and Primed+ conditions (our null hypothesis H_0). A total of 9 and 8 people participated in the Primed and Primed+ conditions respectively.

Methodological Comparison Both procedures were easier to administer than the experiential and conceptual ones. First, the familiarity acquired in the training session reduced the need to explain the interface to the users. On average, each experiment took 30 minutes in the Primed condition and 60 minutes in the Primed+ condition. Neither conditions seemed tiring for users. Second, users in both conditions found the queries easier to understand than users in the Conceptual condition. Finally, users in the Primed+ condition were able to use their experiential query responses as a reference for future responses. These initial experiential queries provided users with a quick feeling for the sequential costs and benefits of using the toolbar.

Quantitative Comparison We conducted a pairwise analysis between the mean utilities in the Conceptual and Primed conditions, and between those in the Conceptual and Primed+ conditions. Using Hotelling's T^2 statistic, we found that the mean utilities between Primed/Primed+ and Conceptual conditions are significantly different ($p < 0.01$ and $p < 0.05$ respectively). Thus, we reject H_0 in both instances. The results of a pairwise analysis using a two-tailed t-test with independent means for each outcome are shown in Figure 5. None of the outcomes individually are significantly different between the new conditions and the Conceptual condition.

From Figure 5, we see that the primed means are generally lower (t score less than 0) than conceptual. In fact, the t scores for the Primed condition (vs. Conceptual) are always lower than the t scores for Experiential (vs. Conceptual). One explanation for this is the fact that the training session in the Primed setting gave users a quick estimate of the costs of searching through and evaluating suggestions

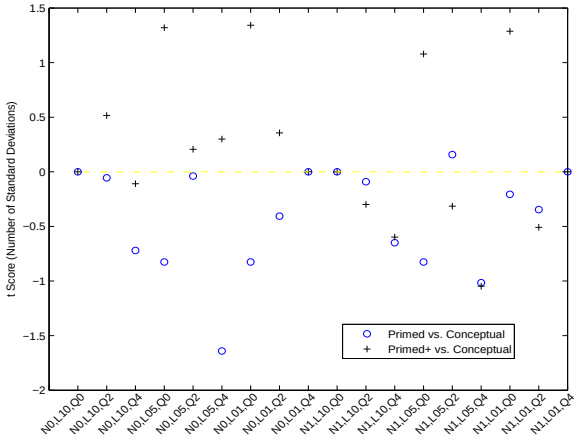


Figure 5: Resulting t scores for each elicited outcome.

in the toolbar—a cost that would otherwise be "unknown" in the Conceptual condition—but that the short experience was insufficient to provide the user with a sense of the *benefits* of help. In both the Primed and Conceptual conditions, the cognitive demand involved in repeated highlighting tasks and the value of help is consistently underestimated. In contrast, the Primed+ condition and its five experiential queries provide a sense of long-term benefits of using the toolbar. Indeed, in Figure 5, the general pattern indicates that Primed+ approaches Experiential. These results support our initial hypothesis that experiential queries enable the user to perceive the full value of adaptive help under realistic circumstances.

6 Conclusions and Future Work

Traditional approaches to utility elicitation do not seem to be effective in assessing the preferences of users for interface customization and adaptive help design, largely due to the lack of experiential assessment. We have proposed a new experiential elicitation procedure that is well-suited to this and related tasks. Our results show that experiential elicitation has several benefits, including: ease of administration from the researcher's perspective; understanding of outcomes from the user's perspective; and helping users appreciate the sequential nature of interaction, often overlooked in traditional, "conceptual" elicitation. We also developed the primed+ procedure to speed up experiential elicitation while maintaining most of its benefits.

In this paper, we focused on eliciting a utility function with three attributes that model savings and processing cost in the context of interface customization. In general, a user's utility function may involve other attributes, depending on the possible customization actions. For example, an adaptive system that hides unused functions causes *disruption* to the user's mental model of the application, but reduces interface *bloat*. By adopting the methodology illustrated in this work, analogous experiments can be devised to ex-

perientially elicit user preferences over these attributes for interface customization.

Future plans include gathering more data to potentially learn a parametric form for the utility function $U(N, L, Q)$. Intuitively, our results suggest a quadratic functional form may explain most preferences, but more data is needed to draw definitive conclusions. We are also interested in examining the extent to which a utility function of this form can be applied more generally to different customization and help tasks. This would allow for the learning of individual user utility models that apply to multiple tasks and even multiple applications. Finally, we are interested in the extent to which lessons in offline elicitation influence the development of online active elicitation and utility assessment strategies, especially the development of behavioral and query response models (e.g., for a POMDP).

Further development of experiential elicitation will require better understanding of which aspects of the outcomes make them experientially different (either better or worse) from a user's conceptual prediction. For example, Figure 4 indicates that, on average, the outcome $n1, l10, q4$ is actually not as good as people think after experiencing it. One explanation is that users did not expect much difficulty in searching for a matching icon ($q4$) when neediness is high. When compared to the mean utility of $n0, l10, q4$, we see that users underestimate the value of help when they are needy. We believe more interesting patterns will unfold in richer domains (i.e., with more attributes and outcomes) and with more data.

References

[1] D. Albrecht, I. Zukerman, and A. Nicholson. Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction*, 8(1-2):5–47, 1998.

[2] K. J. Aström. Optimal control of Markov decision processes with incomplete state estimation. *J. Math. Anal. Appl.*, 10:174–205, 1965.

[3] N. Blaylock and J. Allen. Statistical goal parameter recognition. *ICAPS*, pp.297–304, 2004.

[4] C. Boutilier. A POMDP formulation of preference elicitation problems. *AAAI*, pp.239–246, 2002.

[5] A. Bunt, C. Conati, and J. McGrenere. What role can adaptive support play in an adaptable system? *IUI*, pp.117–124, 2004.

[6] S.K. Card, P.T. Moran, and A. Newell. *Psychology of HCI*. Hillsdale, NJ: Erlbaum, 1980.

[7] U. Chajewska, L. Getoor, J. Norman, and Y. Shahar. Utility elicitation as a classification problem. *UAI*, pp.79–88, 1998.

[8] M. Claypool, P. Le, M. Waseda, and D. Brown. Implicit interest indicators. *IUI*, pp.33–40, 2001.

[9] C. Conati and H. Mclaren. Data-driven refinement of a probabilistic model of user affect. *UM*, pp.40–49, 2005.

[10] G. Foster, P. Langlais, and G. Lapalme. User-friendly text prediction for translators. *EMNLP*, pp.148–155, 2002.

[11] K.Z. Gajos and D. Weld. Preference elicitation for interface optimization. *UIST*, pp.173–182, 2005.

[12] K.Z. Gajos and D.S. Weld. SUPPLE: automatically generating user interfaces. *IUI*, pp.93–100, 2004.

[13] W. Hick. On the rate of gain of information. *J. Experimental Psych.*, 4:11–36, 1952.

[14] A. Hornof, D. Kieras. Cognitive modeling reveals menu search is both random and systematic. *CHI*, pp.107–114, 1997.

[15] E. Horvitz and J. Apacible. Learning and reasoning about interruption. *ICMI*, pp.20–27, 2003.

[16] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, K. Rommelse. The Lumière project: Bayesian user modeling for inferring the goals and needs of software users. *UAI*, pp.256–265, 1998.

[17] E. Horvitz, P. Koch, and J. Apacible. BusyBody: creating and fielding personalized models of the cost of interruption. *CSCW*, pp.507–510, 2004.

[18] C.K. Hsee and J. Zhang. Distinction bias: misprediction and mischoice due to joint evaluation. *Personality and Social Psychology*, 86(5):680–695, 2004.

[19] B. Hui and C. Boutilier. Who's asking for help? A Bayesian approach to intelligent assistance. *IUI*, pp.186–193, 2006.

[20] B. Hui, S. Gustafson, P. Irani, and C. Boutilier. The need for an interaction cost model. *AVI*, pp.458–461, 2008.

[21] B. Hui, S. Liaskos, and J. Mylopoulos. Requirements analysis for customizable software: a Goals-Skills-Preferences framework. *RE*, pp.117–126, 2003.

[22] R. Hyman. Stimulus information as a determinant of reaction time. *J. Experimental Psych.*, 45:188–196, 1953.

[23] D. Kahneman and J. Snell. Predicting utility. In R.M. Hogarth, ed., *Insights in Decision Making*, pp.295–310. Univ. of Chicago Press, 1990.

[24] D. Kahneman and R.H. Thaler. Utility maximization and experienced utility. *Econ. Perspectives*, 20(1):221–234, 2006.

[25] D. Kahneman, P.P. Wakker, and R. Sarin. Back to Bentham? explorations of experienced utility. *Quart. J. Econ.*, 112(2):375–405, 1997.

[26] A. Kapoor, W. Burleson, and R.W. Picard. Automatic prediction of frustration. *Intl. J. Human-Computer Studies*, 65(8):724–736, 2007.

[27] R.L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-Offs*. Wiley, NY, 1976.

[28] N. Lesh. Adaptive goal recognition. *IJCAI*, pp.1208–1214, 1997.

[29] G. Linden, S. Hanks, and N. Lesh. Interactive assessment of user preference models: The automated travel assistant. *UM*, pp.67–78, 1997.

[30] D. Metzler and W.B. Croft. Beyond bags of words: Modeling implicit user preferences in information retrieval. *AAAI*, pp.1646–1649, 2006.

[31] P. Pu, B. Faltings, and M. Torrens. User-involved preference elicitation. *IJCAI Workshop on Configuration*, pp.56–63, 2003.

[32] S. Shearin and H. Lieberman. Intelligent profiling by example. *IUI*, pp.145-152, 2001.

Speeding Up Planning in Markov Decision Processes via Automatically Constructed Abstractions

Alejandro Isaza and **Csaba Szepesvári** and **Vadim Bulitko** and **Russell Greiner**
Department of Computing Science, University of Alberta
Edmonton, Alberta, T6G 2E8, CANADA
{isaza,szepesva,bulitko,greiner}@cs.ualberta.ca

Abstract

In this paper, we consider planning in stochastic shortest path (SSP) problems, a subclass of Markov Decision Problems (MDP). We focus on medium-size problems whose state space can be fully enumerated. This problem has numerous important applications, such as navigation and planning under uncertainty. We propose a new approach for constructing a multi-level hierarchy of progressively simpler abstractions of the original problem. Once computed, the hierarchy can be used to speed up planning by first finding a policy for the most abstract level and then recursively refining it into a solution to the original problem. This approach is fully automated and delivers a speed-up of two orders of magnitude over a state-of-the-art MDP solver on sample problems while returning near-optimal solutions. We also prove theoretical bounds on the loss of solution optimality resulting from the use of abstractions.

1 Introduction and Motivation

We focus on planning in stochastic shortest path problems (the problem of reaching some goal state under uncertainty) when planning time is critical — a situation that arises, for instance, in path planning for agents in commercial video games, where map congestions are modeled as uncertainty of transitions. Another example is path planning for multi-link robotic manipulators, where the uncertainty comes from unmodeled dynamics as well as sensor and actuator noise. More specifically, we consider the problem of finding optimal policies in a sequence of stochastic shortest-path problems (Bertsekas & Tsitsiklis, 1996), where the problems share the same dynamics and transition costs, and differ only in the location of the goal-state.

When the state space underlying the problems is sufficiently large, exact planning methods are unable to deliver a solution within the required time, forcing the user to resort to approximate methods in order to scale to large domains. Exploiting the fact that multiple planning problems share the same dynamics and transition costs, we build an abstracted representation of the shared structure where planning is faster, then map the individual planning problem into the abstract space and derive a solution there. The solution is then refined back into the original space.

In a related problem of path planning under real-time constraints in *deterministic* environments (e.g., Sturtevant, 2007), a particularly successful approach is implemented in the PR LRTS algorithm (Bulitko, Sturtevant, Lu, & Yau, 2007), which builds an abstract state space by partitioning the set of states into cliques (i.e., each state within each cluster is connected to each other state in that cluster with a single action). Each such cluster becomes a single abstract state. Two abstract states are connected by an abstract transition if there is a pair of non-abstract states (one from each abstract state) connected by a single action. The resulting abstract space is smaller and simpler, yet captures some of the structure of the original search problem. Thus, an abstract solution can be used to guide and constrain the search in the original problem, yielding a significant speed-up. Further speed-ups can be obtained by building abstractions on top of abstractions, which creates a hierarchy of progressively smaller abstract search spaces. PR LRTS can then be tuned to meet strict real-time constraints while minimizing solution suboptimality.

Note that state cliques produced by PR LRTS make good abstract states because landing anywhere in such a cluster puts the agent a single action away from any other state in the clique. This also means that the costs of the resulting actions are similar, and that the cost of a single action is negligible compared with the cost of a typical path. Finally, any (optimal) path in the original problem can be closely approximated at the abstract level, as an agent following an (optimal) path has to traverse from cluster to cluster. Since all neighboring clusters are connected in the abstract problem, it is always possible to find a path in the abstract problem that is "close" to the original path.

Given the attractive properties or PR LRTS, it is natural to ask whether the ideas underlying it can be extended to *stochastic* shortest path problems, with *arbitrary* cost structures.

In a stochastic problem, the result of planning is a closed loop policy that assigns actions to states. A successful ab-

straction must be suitable for approximating the execution trace of an optimal policy. Imagine that clustering has been done in some way. The idea is again to have abstract actions that connect neighboring clusters cheaply — that is, the system should not produce expensive connections.

Intuitively, we want to connect one cluster to another if, from any state of the first cluster, we can reliably get to some state of the second cluster at roughly a fixed cost (the same for any state in the first cluster). This way, "simulating" a policy of the original problem becomes possible at a small additional cost (the meaning of simulation will become clear later). This means that a connection between clusters is implemented by a policy with a specific set of initial states that brings the agent from any state of the source cluster to some state of the target cluster. We will use options (Sutton, Precup, & Singh, 1999) for such policies, and choose clusters to allow such policies for any two neighboring clusters. Thus, it is natural to look for clusters of states that allow one to reliably simulate any trajectory from any of the states to any other state.

Finally, we need an extra mechanism, the "goal approach", that deals with the challenge of reaching the base-level goal itself from states that are close to the goal. Thus, our planner first plans in the abstract space to reach the "goal cluster". After arriving at some state of the "goal approach region", the planner then uses the "goal-approach policy" that, with high probability, moves the agent to the goal state itself. These ideas form the core of our algorithm.

The three major contributions of the paper are: (i) a novel theoretical analysis of option-based abstractions, (ii) an effective algorithm for constructing high-quality option-based abstractions, and (iii) experimental results demonstrating that our algorithm performs effectively over a range of problems of varying size and difficulty.

Section 2 formally describes our problem, and provides the theoretical underpinning of our approach. Section 3 then presents our algorithm for automatically building options-based abstractions, and Section 4, our planning algorithm that uses these abstractions. Section 5 empirically evaluates this approach, in terms of both efficiency and effectiveness (suboptimality). Finally, Section 6 summarizes related work.

2 Problem Formulation and Theory

This section formally defines stochastic shortest path problems and the abstractions that we will consider. It also presents a theoretical result that characterizes the relationship between the performance of abstract policies and policies of the original problem.

Definition 1 A *Markov Decision Process* (MDP) is defined by a finite state space $X = \{1, \ldots, n\}$; a finite set of actions $A(x)$ for each state $x \in X$; transition probabilities $p(y|x,a) \in [0,1]$ that correspond to the probability that the next state is y when action a is applied in state x; immediate cost $c(x,a,y) \in \Re$ for all $x, y \in X$ and all $a \in A(x)$ and a discount factor $\gamma \in (0,1]$.

The MDP is undiscounted if $\gamma = 1$. An action $a \in \cup_{x \in X} A(x)$ is called admissible in state x if $a \in A(x)$.

Definition 2 A *(generic) policy* is a mapping that assigns to each history $(x_0, a_0, c_0, \ldots, x_{t-1}, a_{t-1}, c_{t-1}, x_t)$ an action admissible in the most recent state x_t. In general, a mapping that maps possible histories to some set is called a *history dependent mapping*.

Under mild conditions, it suffices to consider only stationary, deterministic policies (Bertsekas & Tsitsiklis, 1996), on which we will focus:

Definition 3 A stationary and deterministic *policy* π is a mapping of states to actions such that $\pi(x) \in A(x)$ holds for any state $x \in X$.

In what follows, we will use "policy" to mean stationary and deterministic policies, unless otherwise mentioned.

The expected cost of policy π when the system starts in state x_0 is $v_\pi(x_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c(X_t, \pi(X_t), X_{t+1})\right]$ where X_t is a Markov chain with $\mathbb{P}\left(X_{t+1} = y | X_t = x\right) = p(y|x, \pi(x))$. The function v_π is called the value-function underlying policy π.

One "solves" an MDP by finding a policy that minimizes the cost from every state, simultaneously. In this paper we deal only with stochastic shortest path problems, a subclass of MDPs. In these MDPs the problem is to get to a goal state with the least cost:

Definition 4 A finite *stochastic shortest path* (SSP) problem is a finite undiscounted MDP that has a special state, called the goal state g, such that $\forall a \in A(g)$, we have $p(g|g,a) = 1$ and $c(g,a,g) = 0$ and the immediate costs for all the other transitions are positive.

Consider a finite SSP (X, A, p, c). Let π be a stationary policy. We say that this policy is *proper* if it reaches the goal state g with probability one, regardless of the initial state. Let $T_\pi : \mathbb{R}^X \to \mathbb{R}^X$ be the policy's evaluation operator:

$$(T_\pi v)(x) = \sum_{y \in X} p(y|x, \pi(x)) \left[c(x, \pi(x), y) + v(y)\right].$$

Bertsekas and Tsitsiklis (1996) prove that T_π is a contraction with respect to a weighted maximum norm, $\|\cdot\|_{w,\infty}$, with some positive weights, $w \in \mathbb{R}^X_+$, where $\|v\|_{w,\infty} = \max_x |v(x)|/w(x)$. In particular, $w(x)$ can be chosen to be the expected number of steps until π reaches the goal state when started from x. The contraction coefficient of T_π, γ_π, satisfies $1/(1-\gamma_\pi) = \max_{x \in X} w(x)$. Thus, $1/(1-\gamma_\pi)$ is the maximum of the expected number of steps to reach the goal state, or in other words, the maximum expected time policy π spends in the MDP (cf. Prop 2.2 in Bertsekas & Tsitsiklis, 1996).

We adopt the notion of options from Sutton et al. (1999):

Definition 5 An *option* is a triple (π, I, ψ), where $I \subset X$ is the set of initial states, π is a (generic) policy that is

defined for histories that start with a state in I and ψ is a history dependent mapping with range $\{0,1\}$, called the terminating condition. We say that the terminating condition fires when $\psi(h_t) = 1$. Let $T > 0$ denote the random time when the terminating condition fires for the first time while following π. (Note that $T = 0$ is not allowed.) We assume that $\mathbb{P}(T < +\infty) = 1$, independent of the initial state when the policy π is started (i.e., the option terminates in finite time with probability one).

As suggested in the introduction, an abstraction is a way to group states and the abstract actions correspond to options:

Definition 6 We say that the MDP $(\tilde{\mathcal{X}}, \tilde{\mathcal{A}}, \tilde{p}, \tilde{c})$ is an *option-based abstraction* of (X, A, p, c), if there exists a mapping, $S : \tilde{\mathcal{X}} \to 2^X$ specifying the states $S(\tilde{x}) \subset X$ that correspond to an abstract state $\tilde{x} \in \tilde{\mathcal{X}}$, a set Π of options abstracting the actions of the MDP and a mapping $\Psi : \cup_{\tilde{x}\in\tilde{\mathcal{X}}} \tilde{\mathcal{A}}(\tilde{x}) \to \Pi$ such that for any $\tilde{a} \in \tilde{\mathcal{A}}(\tilde{x})$, if $\Psi(\tilde{a}) = (I, \pi, \psi)$ then $S(\tilde{x}) \subset I$.[1]

Henceforth we will use "abstraction" instead of "option-based abstraction" and will call $(\tilde{\mathcal{X}}, \tilde{\mathcal{A}}, \tilde{p}, \tilde{c})$ the "abstract MDP", $\tilde{\mathcal{X}}$ the set of abstract states, $\tilde{\mathcal{A}}$ the set of abstract actions, etc. Notationally, we call (X, A, p, c) the *ground level* MDP, and we will identify quantities related to the abstract MDP by using a tilde ($\tilde{}$). For simplicity, we will identify the abstract actions with their corresponding options. In particular, we will call $\tilde{a}$ both an abstract action and an option, depending on the context.

In the following, we will assume that $\{S(\tilde{x}) \,|\, \tilde{x} \in \tilde{\mathcal{X}}\}$ is a partition of X; we can then let $\tilde{x} : X \to \tilde{\mathcal{X}}$ denote the (unique) abstract state that includes x: $\tilde{x}(x) \in \tilde{\mathcal{X}}$ such that $x \in S(\tilde{x}(x))$, and say that $(\tilde{\mathcal{X}}, S)$ is an *aggregation of the states* in X. We also define $S(x) = S(\tilde{x}(x))$ as the set of states in X that are in the same partition with x.

The restriction on Ψ in the above definition ensures that the execution of any policy $\tilde{\pi}$ in the abstract MDP is well-defined and proceeds as follows. Initially, there is no active option. In general, whenever there is no active option, we look up the abstract state $\tilde{x} = \tilde{x}(x)$ based on the current state x and activate the option $\Psi(\tilde{\pi}(\tilde{x}))$. When there is an active option, the option remains active until the corresponding terminating condition fires. When an option is active, the option's policy selects the actions in the ground level MDP. This way a policy $\tilde{\pi}$ in the abstract MDP *induces* a policy in the ground level MDP.

Our goal now is to characterize what makes an abstraction accurate. The following theoretical analysis is novel as it considers abstractions where the action set is changed. In particular, the action set can potentially be reduced and the abstract actions can be options. To our knowledge, such options-based abstractions have not been analyzed previously; the closest results are probably Theorem 2 of Kim and Dean (2003) and Theorem 4 of Dean, Givan, and Leach (1997). The proof is rather technical and is given in the extended version of our paper (Isaza, Szepesvári, Bulitko, & Greiner, 2008).

Consider a proper policy π of the ground level MDP. We want abstractions such that one can always find a policy in the abstract MDP $(\tilde{\mathcal{X}}, \tilde{\mathcal{A}}, \tilde{p}, \tilde{c})$ that approximates π well, no matter how π was chosen. Clearly, this depends on how the action set $\tilde{\mathcal{A}}$ and the corresponding transitions and costs are defined in the abstract MDP. Quantifying this requires a few definitions: Let $\tilde{p}_\pi(\tilde{x}, \tilde{y})$ be the probability of landing in some state of $S(\tilde{y})$ when following policy π until it leaves the states of $S(\tilde{x})$, when the initial state is selected at random from the states of $S(\tilde{x})$ based on the distribution $\mu_{S(\tilde{x})}$. Let $\tilde{c}_\pi(\tilde{x})$ denote the corresponding expected "immediate" cost. Now pick a proper policy $\tilde{\pi}$ of the abstract MDP. Let $\tilde{w}$ be the weight vector that makes $T_{\tilde{\pi}}$ a contraction in the abstract MDP. Further, define $\tilde{p}_{\tilde{\pi}}(\tilde{x}, \tilde{y}) = \tilde{p}(\tilde{y}|\tilde{x}, \tilde{\pi}(\tilde{x}))$ and $\tilde{c}_{\tilde{\pi}}(\tilde{x}) = \sum_{\tilde{y}\in\tilde{\mathcal{X}}} \tilde{p}_{\tilde{\pi}}(\tilde{x}, \tilde{y})\tilde{c}_{\tilde{\pi}}(\tilde{x}, \tilde{y})$ and the mixed ℓ^1/ℓ^∞ norm $\|\cdot\|_{\tilde{w},1/\infty}$:

$$\|\tilde{p}_1 - \tilde{p}_2\|_{\tilde{w},1/\infty} = \max_{\tilde{x}\in\tilde{\mathcal{X}}} \sum_{\tilde{y}\in\tilde{\mathcal{X}}} |\tilde{p}_1(\tilde{x}, \tilde{y}) - \tilde{p}_2(\tilde{x}, \tilde{y})| \frac{\tilde{w}(\tilde{y})}{\tilde{w}(\tilde{x})}.$$

Let

$$\varepsilon_{\pi,\tilde{\pi}} = \|\tilde{c}_\pi - \tilde{c}_{\tilde{\pi}}\|_{\tilde{w},\infty} + c_{\max} \|\tilde{p}_\pi - \tilde{p}_{\tilde{\pi}}\|_{\tilde{w},1/\infty}, \qquad (1)$$

where $c_{\max}$ is the maximum of the immediate costs in the ground level MDP. Hence, $\varepsilon_{\pi,\tilde{\pi}}$ measures how well the costs and the transition probabilities induced by π "after state aggregation" match those of $\tilde{\pi}$. Introduce $c(x, \pi)$ as the expected total cost incurred, conditioned on that policy π starting in state x and stopping when it exits $S(x)$. Further, introduce $p(\tilde{y}|x, \pi)$ as the probability that, given that policy π is started in state x, when it exits $S(x)$ it enters $S(\tilde{y})$ ($\tilde{y} \neq \tilde{x}(x)$). Now fix an abstract state $\tilde{x} \in \tilde{\mathcal{X}}$. If the costs $\{c(x, \pi)\}_{x\in S(\tilde{x})}$ and probabilities $\{p(\tilde{y}|x, \pi)\}_{x\in S(\tilde{x})}$, $\tilde{y} \neq \tilde{x}$, have a small range then we can model closely the behavior of π locally at $S(\tilde{x})$ by introducing an option with initial states in $S(\tilde{x})$ which mimics the "expected" behavior of π as it leaves $S(\tilde{x})$, assuming, say, that the initial state in $S(\tilde{x})$ is selected at random according to the distribution $\mu_{S(\tilde{x})}$. If we do so for all abstract states $\tilde{x} \in \tilde{\mathcal{X}}$ then we can make sure that $\min_\pi \varepsilon_{\pi,\tilde{\pi}}$ is small. If the above range conditions hold for all policies π of the ground level MDP and all abstract states $\tilde{x} \in \tilde{\mathcal{X}}$ then by introducing a sufficiently large number of abstract actions it is possible to keep $\max_\pi \min_{\tilde{\pi}} \varepsilon_{\pi,\tilde{\pi}}$ small. Further, notice that $\max_\pi p(\tilde{y}|x, \pi)$ is zero unless there exists a transition from some state of $S(x)$ to some state of $S(\tilde{y})$, in which case we say that $S(x)$ is connected to $S(\tilde{y})$. Hence, no abstract action is needed "between" $\tilde{x}$ and $\tilde{y}$, unless $S(\tilde{x})$ is connected in the ground level MDP to $S(\tilde{y})$.

Define $\tilde{T}_\pi : B(\tilde{X}) \to B(\tilde{X})$, $(\tilde{T}_\pi \tilde{v})(\tilde{x}) = \tilde{c}_\pi(\tilde{x}) + \sum_{\tilde{y}} \tilde{p}_{\tilde{\pi}}(\tilde{x}, \tilde{y})\tilde{v}(\tilde{y})$. Since π is proper in the ground level MDP, it is not difficult to show that $\tilde{T}_\pi$ is a contraction with respect to an appropriately defined weighted supremum norm.

The next result gives a bound on the difference of value functions of π and $\tilde{\pi}$ in terms of $\varepsilon_{\pi,\tilde{\pi}}$:

[1] 2^X denotes the power set of X: the set of all subsets of X.

Theorem 1 Let π be a proper policy in the ground level MDP and let $\tilde{\pi}$ be a proper policy in the abstract MDP. Let w_π (resp., $\tilde{w}_\pi$) be the weight vector that makes T_π (resp., $\tilde{T}_\pi$) a contraction and let the corresponding contraction factor be γ_π (resp., $\tilde{\gamma}_\pi$). Let v_π be the value function of π and $\tilde{v}_{\tilde{\pi}}$ be the value function of $\tilde{\pi}$. Then

$$\|v_\pi - E\tilde{v}_{\tilde{\pi}}\|_{w,\infty} \leq \frac{\|Av_\pi - v_\pi\|_{w,\infty}}{1-\gamma_\pi} + \lambda_\pi \frac{\varepsilon_{\pi,\tilde{\pi}}}{1-\tilde{\gamma}_\pi},$$

where the operator E extends functions defined over $\tilde{\mathcal{X}}$ to functions defined over X in a piecewise constant manner: $E : B(\tilde{\mathcal{X}}) \to B(X)$, $(E\tilde{v})(x) = \tilde{v}(\tilde{x}(x))$, and $A : B(X) \to B(X)$ is the aggregation operator defined by

$$(AV)(x) = \sum_{z \in S(x)} \mu_{S(x)}(z) V(z),$$

and $\lambda_\pi = \max_{x \in X} \tilde{w}_\pi(\tilde{x}(x))/w_\pi(x)$.

The factor λ_π measures how many more steps are needed to reach the goal if the execution of policy π is modified such that, whenever the policy enters a new cluster $\tilde{x}$, the state gets perturbed, by choosing a random state according to $\mu_{S(\tilde{x})}$.

The theorem provides a bound on the difference between the value function of a ground-level policy π and the value function of an abstract policy when its value function is extended to the ground-level states. The bound has two terms: The first bounds the loss due to state abstraction, while the second bounds the loss due to action abstraction. When a similar range condition holds for the abstract actions, too, then it is possible to bound the difference between the value function of the policy induced in the ground level MDP by $\tilde{\pi}$ and $E\tilde{v}_{\tilde{\pi}}$, yielding a difference on the value functions of π and the policy induced by $\tilde{\pi}$. Isaza et al. (2008) provides further details.

If we apply this result to an optimal policy π^* of the ground level MDP, we immediately get a bound on the quality of the abstraction. We may conclude then that the quality of abstraction is determined by the following factors: (i) whether states with different optimal values are aggregated; (ii) whether the random perturbation described in the previous paragraph can increase the number of steps to the goal substantially; and (iii) whether the immediate costs $\tilde{c}_{\pi^*}$ and transition probabilities $\tilde{p}_{\pi^*}$ can be matched in the abstract MDP.

Since we want to build abstractions that work independently of where the goal is placed, the knowledge of the optimal policy with respect to a particular goal cannot be exploited when constructing the abstractions. In order to prevent large errors due to (i) and (ii), we restrict aggregation such that only a few states are grouped together. This makes the job of creating an aggregation easier. Fortunately, we can achieve higher compression by adding additional layers of abstractions. We can address (iii) by creating a sufficiently large number of abstract actions. Here, we use the simplifying assumption that we only create abstract actions that bring the agent from some cluster of states to some neighboring cluster. These can serve as a "basis" for matching any complex next-state distribution over the clusters by choosing an appropriate stochastic policy in the abstract MDP. We also want to ensure that the initial state within a cluster has a small influence on the probability of transitioning to some neighboring cluster and the associated costs. We use two constants, ε and μ, to bound the amount of variation with respect to initial states; note this allows us to control the difference between the value function of a policy induced in the ground level MDP by some abstract policy $\tilde{\pi}$ and the extension of the value function of $\tilde{\pi}$ defined in the abstract MDP to the ground level states, $E\tilde{v}_{\tilde{\pi}}$. This is necessary to ensure that a good policy in the abstract MDP produces a good policy in the ground-level MDP, ultimately assuring that the optimal policy of the abstract MDP will give rise to a close to optimal policy in the ground-level MDP. The resulting procedure is described in the next section.

3 Abstracting an SSP

This section describes our algorithm **BuildAbstraction** for automatically building options-based abstractions. These abstractions are goal-independent and thus apply to a series of SSPs that share the state space and transition dynamics. The process consists of four main steps (Figure 1): (1) *Cluster* proposes candidates for abstract states; (2) *GenerateLinkCandidates* proposes candidates for abstract actions (or "links"); (3) *Repair* validates and, if necessary, repairs the links in order to satisfy the so-called (ε, μ)-connectivity property (the formal definition is given later) and *Prune* discards excessive links.

Once an abstraction is built, we use a special-purpose planning procedure (described in Section 4) to solve specific SSPs. The rest of this section describes the four steps of our **BuildAbstraction** algorithm in detail.

Step 1: *Cluster.* A straightforward cluster-er will cluster a state with some of its immediate neighbors. Unfortunately, this approach may group states with diverging trajectories (the trajectories from one state can differ from those of the other state). By looking for the peers of a state (predecessors of its successors, line 2, Figure 1) we hope to find a peer whose trajectories are similar to the trajectories of the first state. Note that the clustering routine creates minimal clusters. This is advantageous as it means the subsequent steps, which connect clusters, is more likely to succeed. Unfortunately, it also means relatively low reduction in the number of states. Several layers of abstractions can help increase this reduction.

Step 2: *Generate Link Candidates.* After forming the initial clusters (i.e., the initial abstract states), **BuildAbstraction** generates candidates for abstract actions. One approach is simply to propose abstract actions for *all* pairs of abstract states, in the hope that only important ones will remain after pruning. We use a less expensive strategy and propose abstract action candidates only for "nearby" clusters (line 8). For each such pair we add two candidate links: one in the forward and another in the backward direction — this heuristic quickly generates reasonable link candidates. We typically use $k = 1$. Our experiments confirm this is

BuildAbstraction(k, p, M) // M – ground level MDP

–Cluster–

1 **for** each unmarked ground state x **do**
2 Find $P(x)$, all the predecessors of successors of x
3 Find $y \in P(x)$ that has the most successors in common with x
4 Add $\tilde{x}$ to $\tilde{X}$ with $S(\tilde{x}) = \{x, y\}$
5 Mark states $\{x, y\}$
6 **end for**

–GenerateLinkCandidates–

7 repairQ $\leftarrow \emptyset$
8 **for** every $\tilde{x}, \tilde{y} \in \tilde{X}$, where any state in $S(\tilde{y})$ is within k ground transitions of some state in $S(\tilde{x})$ **do**
9 repairQ $\leftarrow$ repairQ $\cup \{(\tilde{x}, \tilde{y}),\ (\tilde{y}, \tilde{x})\}$
10 **end for**

–Repair–

11 **while** repairQ $\neq \emptyset$ **do**
12 $(\tilde{x}, \tilde{y}) \leftarrow$ pop an element from repairQ
13 set up an SSP, S, with domain $R \subset X$ where $S(\tilde{x}) \cup S(\tilde{y}) \subset R$ with states in $S(\tilde{y})$ as goals
14 attempt to find an optimal policy π_S in S with IPS
15 **if** no policy found **then**
16 **continue**
17 **else if** π_S does not meet the (ε, μ) conditions **then**
18 split the cluster adding both parts to repairQ
19 **else**
20 add $\tilde{a}$ to $\tilde{A}(\tilde{x})$ with $\Psi(\tilde{a}) = (S(\tilde{x}), \pi_S, \mathbb{I}_{S(\tilde{y})})$ [2]
21 set $\tilde{c}(\tilde{x}, \tilde{a})$ to be the expected cost of executing $\tilde{a}$ from a random state of $\tilde{x}$
22 set $\tilde{p}(\tilde{y}|\tilde{x}, \tilde{a}) = 1$, $\tilde{p}(\tilde{y}'|\tilde{x}, \tilde{a}) = 0$ for $\tilde{y}' \neq \tilde{y}$.
23 **end if**
24 **end while**

–Prune–

25 **for** each state $\tilde{x}$ **do**
26 find $\tilde{A}^*(\tilde{x}) = \{\tilde{a}_1, \ldots, \tilde{a}_m\}$, all abstract actions that connect clusters that are neighbors in M
27 order $\tilde{A}(\tilde{x}) \setminus \tilde{A}^*(\tilde{x})$ to create $[\tilde{a}_{m+1}, \ldots, \tilde{a}_n]$ such that $\tilde{c}(\tilde{x}, \tilde{a}_i) \leq \tilde{c}(\tilde{x}, \tilde{a}_{i+1})$, $i = m+1, \ldots, n-1$
28 let $\tilde{A}(\tilde{x}) = \{\tilde{a}_1, \ldots, \tilde{a}_p\}$
29 **end for**
30 **return** $(\tilde{X}, \tilde{A}, \tilde{p}, \tilde{c})$.

Figure 1: The abstraction algorithm.

sufficient; increasing k results in slightly better quality, but slower running times when solving the planning problems.

Step 3: *Repair.* For each candidate abstract action connecting abstract states $\tilde{x}$ and $\tilde{y}$, we first need to derive an option that, starting in any state in cluster $\tilde{x}$ leads the agent to some state in cluster $\tilde{y}$ with a minimum total expected cost. We derive this option by setting up a shortest path problem S, whose domain includes $S(\tilde{x})$ and $S(\tilde{y})$. We set the domain of S to be sufficiently large that a policy within this domain can reliably take the agent from any state of $S(\tilde{x})$ to some state of $S(\tilde{y})$. **BuildAbstraction** builds this domain by performing a breadth-first search from $S(\tilde{y})$, proceeding backwards along the transitions, stopping at depth $D + m$, where D is the search depth from $S(\tilde{y})$ and m is the margin to leave after all states of $S(\tilde{x})$ were added to the domain. If there is any state of $S(\tilde{x})$ that was not included at depth D, the *Repair* routine reports 'no solution'. The transitions, actions and costs of S are inherited from the MDP M. We also add a new terminating state, which is the destination of transitions leaving the region — i.e., those transitions are redirected to this new terminal, with a transition cost that exceeds the maximum of the total expected costs of the ground level MDP. The high cost discourages the solutions to enter the extra terminating state. The optimal solution to S is obtained by using the *Improved Prioritized Sweeping* (IPS) algorithm of McMahan and Gordon (2005), (line 14). We selected this algorithm based on its excellent performance and known optimality properties (IPS reduces to Dijkstra's method in deterministic problems). The resulting policy π is checked against (ε, μ)-connectivity, defined as follows: we first compute the expected total cost of reaching some states in $S(\tilde{y})$ for all states of $S(\tilde{x})$; let the resulting costs be $c(x, \pi)$. Similarly, we compute the probabilities $p(S(\tilde{y})|x, \pi)$ for every $x \in S(\tilde{x})$. Then we check if $\max_{x,x' \in S(\tilde{x})} |c(x, \pi) - c(x', \pi)| \leq \varepsilon$ and $\max_{x,x' \in S(\tilde{x})} |p(S(\tilde{y})|x, \pi) - p(S(\tilde{y})|x', \pi)| \leq \mu$ both hold. If these constraints are met, a new abstract action is created and is added to the set of admissible actions at $\tilde{x}$ and the policy is stored as the option corresponding to this new abstract action (lines 20–22). Otherwise, the cluster is split (since every cluster has two states, this is trivial) and the appropriate link candidates are added to the repair queue so that no link between potentially connected clusters is missed.

Step 4: *Prune.* After step 3, we have an abstract SSP whose abstract states are (ε, μ)-connected. However, our abstract action generation mechanism may produce too many actions, which may slow down the planning algorithm (see Section 4). We address this problem using a pruning step that leaves only the "critical" and cheapest abstract actions. An action is "critical" if it connects clusters that are connected at the ground level with a single transition; these actions are important to keep the structure of the ground level MDP. We also keep the cheapest abstract actions as they are likely to help achieve high quality solutions. The "pruning parameter", p, specifies the total number of actions to keep. (If p is smaller or equal than the number of ground actions, then only the "critical" actions are kept.)

BuildAbstraction runs in time linear in the size of the input MDP, as every step is restricted to some fixed size neighborhood of some state (i.e., every step is local). Further, employing a suitable data structure, the memory requirements can also be kept linear in the size of the input. These properties are important when scaling up to realistic, real-world problem sizes.

4 Planning with an Abstraction

After building an abstraction, we can use it to solve particular SSP problems. When we specify a new goal, our abstraction planner, **AbsPlanner**, then creates a *goal-approach region* in the abstract MDP that includes the goal and is large enough to include all states of the cluster containing the goal. **AbsPlanner** builds this region by starting with the ground goal and adding states and transitions in a breadth-first fashion to a certain depth, proceeding backwards along the transitions, stopping only after adding all states of the goal-cluster. After building the region, **AbsPlanner** produces an SSP. The domain of this

[2] Here $\mathbb{I}_S$ is the characteristic function of S: $\mathbb{I}_S(x) = 1$ iff $x \in S$ and $\mathbb{I}_S(x) = 0$ otherwise.

SSP includes the states found in the breadth-first search, and also a new terminal state that becomes the destination of transitions leaving the region — i.e., those transitions are redirected to this new terminal, with a high transition cost. All other costs and transitions of this SSP are inherited from the ground level MDP. **AbsPlanner** uses IPS to solve the local MDP, and saves the resulting *goal-approach policy*. It then solves the abstract MDP, where the goal cluster is set as the goal. When executing the resulting policy $\tilde{\pi}$, **AbsPlanner** proceeds normally until reaching a state of the goal-approach region; it then switches to the goal-approach policy, which it follows until reaching the goal or leaving the region. When this latter event happens and the state is x, execution switches to the option $\tilde{\pi}(\tilde{x}(x))$.

When using multiple levels of abstraction, **AbsPlanner**'s execution follows a recursive, hierarchical strategy. Note that the size of the goal-approach region is independent of the size of the MDP. Thus, the planning time will depend on the size of the top-level abstract MDP. For an MDP of size n, by using $\log n$ levels of hierarchy, in theory it is then possible to achieve planning times that scale with $O(\log n)$. However, depending on the problem, it might be hard to guarantee high quality solutions when using many levels of abstraction. Furthermore, in practice (over the problems used in our tests), the computation time is dominated by the time needed to set up and solve the goal-approach SSPs, which is required for even one layer of abstraction. This is partly because our abstractions result in *deterministic* shortest path problems, whose solutions can be found significantly faster than those of stochastic problems.

5 Empirical Evaluation

This section summarizes our empirical evaluation of this approach, in terms of the quality (suboptimality) of the solutions and the solution times. Here we report the trade-offs of using different levels of abstraction as well as the dependence on the "stochasticity" of the transitions. (Note that stochasticity makes it difficult to build abstractions.) We also tested the performance of the algorithm on more practical problems. In addition to the results presented here, we conducted extensive experiments, studying the trade off between solution quality and solution time as a function of the various parameters of our algorithm (e.g., the values of p, k, or the number of abstraction levels), the scaling behavior of our algorithm in terms of its resource usage, the quality of solutions and the solution time. These results, appearing in (Isaza et al., 2008), confirm that the algorithm is robust to the choices of its parameters and scales as expected by increasing problem sizes.

We run experiments over three domains: noisy gridworlds, a "river" and congested game maps. The gridworlds are empty and have four actions: *up*, *down*, *left*, and *right*, each with cost 1. The probability that an action leaded to the expected position (e.g., the action *up* moves the agent up one cell) is 0.7, while the probability of reaching any of the other three adjacent cells is 0.1.

The river is similar to the gridworld: its dimensions are $w \times h$, but there is a current flowing from left to right and a fork corresponding to a line connecting the points $(w/2, h/2)$ and $(w, h/2)$.[3] The flow is represented by modifying both the cost structure and the transition probabilities of the actions: action *forward* costs 1, *backward* costs 5, *diagonally-up-and-forward* and *diagonally-down-and-forward* each cost $\sqrt{2}$. These actions are also stochastic: For the *backward* action, the probabilities are 0.7 for going back and 0.1 for each of the other actions. For the other three actions, the anticipated move occurs with probability 0.6 and the other moves except *backwards* occur each with probability 0.2, and *backwards* has probability 0. We include the river domain to determine whether our system can deal with non-uniform structures and because the fork complicates the task of creating abstractions. We empirically found the time to build abstractions for the n-state gridworld was close to $n/100$ seconds, and around $n/50$ for an n-state river domain. The build time for the maps, using $k = 1$, was between 75 and 100 seconds.[4]

The congested game maps are again similar to gridworlds, but with obstacles and with transitions probabilities that depend on the congestion. The obstacle layout comes from commercial game maps, and the stochastic dynamics simulate what happens if multiple units traverse the same map: in narrow passages, the units to become congested, which means an agent trying to traverse such a passage is likely to be blocked. We model this by modifying each action by including a probability that the action will "fail" and cause the agent to stay at the same position. This "failure probability" depends on the position on the game map, calculated by simulating many units randomly traversing the game maps and measuring the average occupation of the individual cells, then turning the occupation numbers into probabilities. The optimal policy of an agent in a congested game map will then try to avoid narrow passages, since the higher probability of traffic congestion in such regions means an agent takes much longer to get through those regions.

The baseline performance measures are obtained by running the state-of-the-art SSP solver algorithm IPS. For each study, we generate the abstraction and then use it to solve 1,000 problems, whose start and goal locations are selected uniformly at random. For each problem we measure the solution time in seconds and the total solution cost for both IPS and our method, then compute the geometric average of the individual suboptimalities and the individual solution time ratios.

5.1 Abstraction level trade-offs

We used a 100×100 gridworld to analyze the trade-offs of different abstraction levels, with several different parameter configurations. We say a configuration is "dominant" if it was a Pareto optimal — i.e., if no other configuration is better in both time and suboptimality.

Figure 2 presents properties of the dominant configurations

[3] See (Isaza et al., 2008) for more details, including relevant pictures.

[4] We ran all experiments on a 2GHz AMD Opteron(tm) Processor with 4GB of RAM running Linux with kernel 2.6.18.

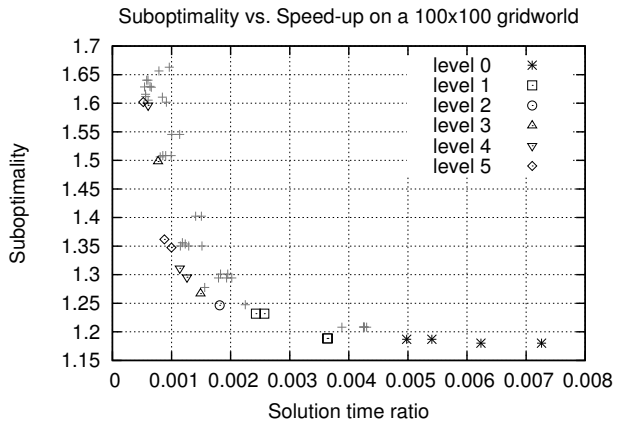


Figure 2: Subobtimality versus the solution time ratio as compared to IPS for different parameter configurations. The dominant configurations are shown for different levels of abstraction.

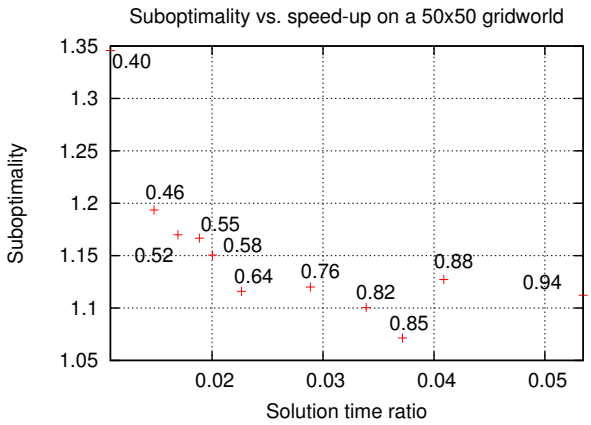


Figure 3: Subobtimality versus the solution time ratio as compared to IPS for different values of P.

for various abstraction levels. We see that using a smaller number of abstractions required more time but produced better solutions (i.e., lower suboptimality), and higher levels of abstractions required less solution time but produced inferior solutions (i.e., increased suboptimality). Note that there are dominant configurations for every level of abstraction, from 0 to 5.

We obtain a "level 0" abstraction by converting the given ground-level SSP to *deterministic* shortest path problem with the same states. (Recall that our abstraction process abstracts the state space and produces a *deterministic* SSP; here we just used the original state space.) Figure 2 shows that this transformation provides solutions whose quality is slightly inferior to the original problem, but it finds this solution significantly faster (e.g., in 0.005 to 0.0073 of the time). We also see that these "level 0" solutions are superior to those based on higher abstraction levels, but one can obtain these level-i solutions in yet less time.

5.2 Sensitivity to Stochasticity of the Dynamics

As the environment becomes noisier, it becomes more difficult to construct a high quality abstraction. This section quantifies how the solution quality and construction time relate to noise in the dynamics. In general, we consider an action "successful" if the agent moves to the appropriate direction; our gridworld model set the success probability to $P = 0.7$, leaving a probability of $(1 - P)/3$ to moving in each of the other three directions. Here, we vary the value of P. All of these experiments use a 50×50 gridworld with $k = 2$ and $p = 4$ (which means we keep only the "critical" actions; see Section 3).

Figure 3 plots the suboptimality and the speed-up of finding a solution using our method, as compared to IPS, for different values of P. We see that our method loses optimality as the dynamics becomes noisier (i.e., when P gets smaller). This is because our abstract actions, trying to move the agent from one abstract state to the next will fail with higher probability for noisier dynamics. Note that the advantage of our method, in terms of planning time, becomes larger with increased stochasticity. This is because our abstractions are deterministic and planning in a deterministic system is much faster than planning with a stochastic system.

Figure 4, plotting the absolute values of cost and time for both our method and IPS, provides another insight: It shows that for increasing stochasticity both methods are slowed down, but our method can cope better with this situation. This figure also confirms that this leads to a loss in solution quality.

For our method the typical parameters produce a suboptimality of around 1.4 for the river, and around 1.25 for the gridworld domain. The speed-up for the gridworld is around 30, while for the river it is around 800.

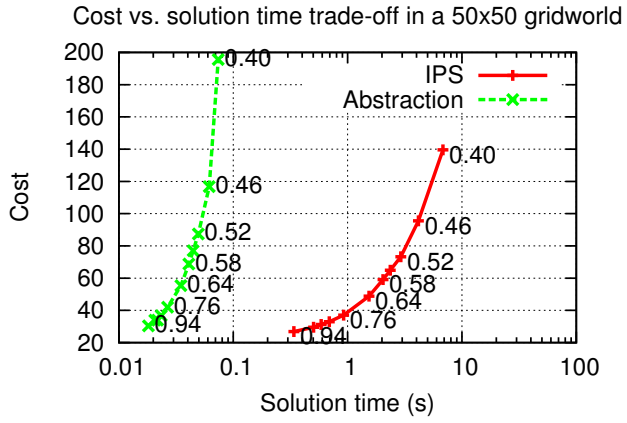


Figure 4: Cost versus solution time for IPS and abstraction at different values of P.

5.3 Congested Game Maps

To test the performance of our approach in a more practical application, we used maps modeled after game environments from commercial video games. We first created simplified gridworlds that resemble some levels from a

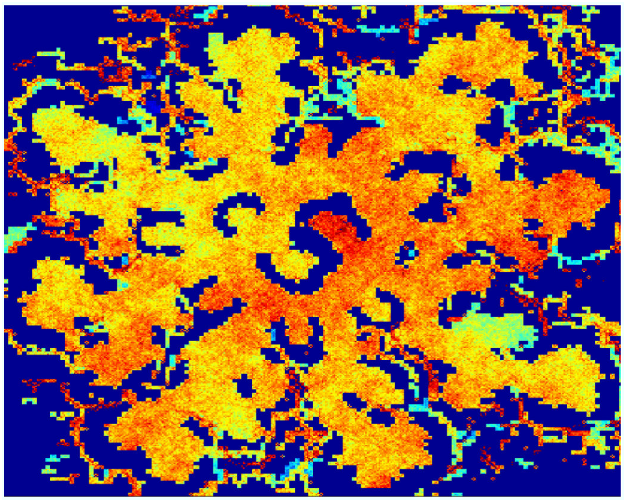


Figure 5: A congested game map. Darker/redder color refers to high congestion. Dark blue regions are impassable obstacles.

popular role-playing and real-time-strategy game. We then converted the gridworlds into congested maps as described earlier. This produced maps with state space sizes of 6176 (BG1), 5672 (BG2), 5852 (BG3), 20249 (WC1) and 9848 (WC2). Figure 5 provides one such map, where each state's color indicates the associated congestion: warmer/redder colors indicates high congestion (i.e., low probability of success P) while colder/bluer colors indicates low congestion (i.e., high value of P). Very dark blue indicates impassable obstacles. We see that many of the states in cluttered regions are highly congested and should therefore be avoided.

Figure 6 shows the solution time and the solution suboptimalities for both our method and IPS, for two maps from WarCraft (Blizzard Entertainment, 2002) and three maps from Baldur's Gate (BioWare Corp., 1998), including Figure 5, using only a single layer of abstraction. We see that our approach is indeed successful in speeding up the planning process, while keeping the quality of the resulting solutions high.

6 Related Work

Due to space constraints we review only the most relevant work; references to other related works can be found in the extensive bibliography lists of the cited works. Dean et al. (1997) introduced the notion of ε-homogeneous partitions and analyzed its properties, but without giving explicit loss bounds. Kim and Dean (2003) developed some loss bounds. Their Theorem 2 can be strengthened with our proof method to $\|v^* - v_P^*\|_\infty \leq \|Tv_P^* - v_P^*\|_\infty / (1-\gamma)$ (using our notation), basically dropping the first term in their bound. Here v^* is the optimal value function in the original MDP, v_P^* is the optimal value function of the aggregated MDP extended back to the original state space in a piecewise constant manner and T is the Bellman-optimality operator in the original MDP. This bound is problematic as it does not show how the quality of partitions influences the loss. Our bound improves on this bound in this respect, and also by extending it to the case when the abstract actions correspond to options. While Asadi and Huber (2004) also considered such options-based abstractions, they assume that the abstract actions (options) are given externally (possibly by specifying goal states for each of them) and they do not develop bounds. In a number of subsequent papers, the authors refined their methods. In particular, they became increasingly focussed on learning problems. For example, in the recent follow-up work, Asadi and Huber (2007) provide a method to learn an abstract hierarchical representation that uses state aggregation and options. Since they are interested in skill transfer through a series of related problems that can *differ* in their cost structure, they introduce a heuristic to discover subgoals based on bottleneck states. They learn options for achieving the discovered subgoals and introduce a partitioning that respects the learned options (in the clusters typically there are many states). The success of the approach relies critically on the existence of meaningful bottleneck states. This leads to a granularity issue: identifying the bottleneck states requires computing a statistic for each state visited, meaning bottlenecks will not be pronounced if resolution is increased in narrow pathways. Nevertheless, the approach has been successfully tested in a non-trivial domain of 20,000 states.

Hauskrecht, Meuleau, Kaelbling, Dean, and Boutilier (1998) introduce a method that also uses options, but the abstract states correspond to boundary states of regions. The regions are assumed to be given *a priori*. The idea is similar to using bottleneck states. In contrast to that work, we do not assume any prior knowledge, but construct the abstractions completely autonomously. Further, we deal with undiscounted SSPs, while Hauskrecht et al. (1998) dealt with discounted MDPs (but this difference is probably not crucial).

7 Discussion and Future Directions

In the approach presented, options serve as closed-loop abstract actions. Another way to use an abstract solution would be to use the abstract value function to guide local search initiated from the current state. These ideas has proven successful in pattern-database research where the cost of an optimal solution of an abstract problem is used as a powerful heuristic for the original problem. Such a procedure has the potential to improve solution quality, while keeping low the cost of the planning steps interleaved with execution. Another idea is to use the abstraction to select the amount of such local search (i.e., the depth of the rollouts); these ideas has proven successful in deterministic environments (Bulitko, Björnsson, Luštrek, Schaeffer, & Sigmundarson, 2007; Bulitko, Luštrek, Schaeffer, Björnsson, & Sigmundarson, 2008).

Presently, our abstractions are deterministic. This suggests two avenues for future work. First, applying advanced heuristic search methods to such abstractions may lead to performance gains. Second, in highly stochastic domains, the abstraction's determinism may lead to a poor quality of solution, as the cost of ensuring arrival at an abstract

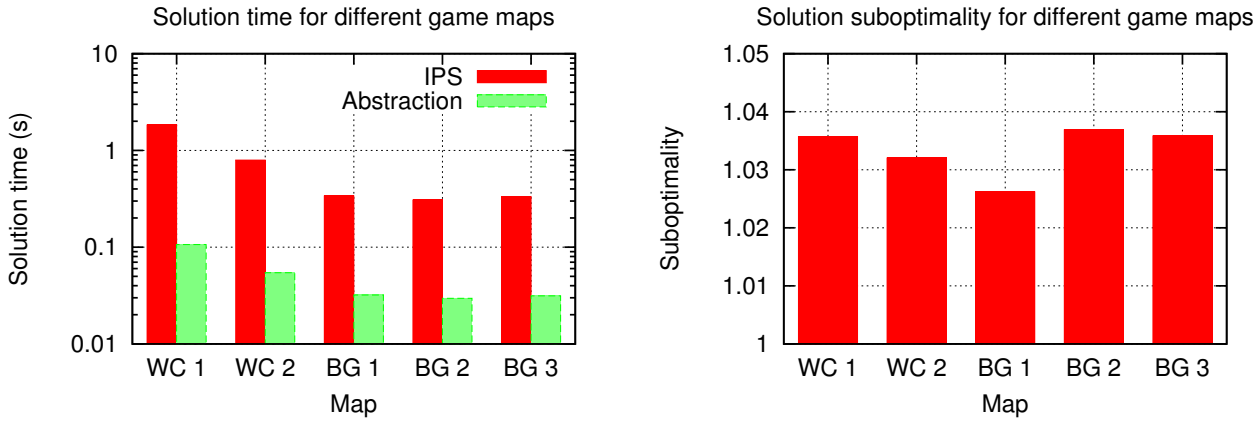


Figure 6: Solution times (left) and suboptimalities (right) for several game maps.

state with certainty (or very high probability) can lead to very conservative and costly paths. Thus, it would be of interest to investigate *stochastic* abstractions. One idea is to modify the way abstract actions are defined: When planning to connect to abstract states after a solution of the local SSP is found, with a little extra work we can compute the probabilities of reaching various neighboring abstract states under the policy found when the policy leaves the region of interest.

Yet another avenue for future work would be to move from a state-based problem formulation to a feature-based one, assuming that the features describe the states. The challenge is to design an algorithm that can construct an abstraction without enumerating all the states, as ours currently does. Although this paper has not attempted to address this problem, we believe that the approach proposed here (i.e., incremental clustering and defining options by solving local planning problems) is applicable.

Finally, although the present paper dealt only with undiscounted, stochastic shortest path problems, the approach can be extended to work for *discounted* problems. This holds because a discounted problem can always be viewed as an undiscounted stochastic shortest path problem where every time step a transition is made to some terminal state with probability $1 - \gamma$, where $0 < \gamma < 1$ is the discount factor.

8 Conclusions

This paper has explored ways to speed up planning in SSP problems via goal-independent state and action abstraction. We strengthen existing theoretical results, then provide an algorithm for building abstraction hierarchies automatically. Finally, we empirically demonstrate the advantages of this approach by showing that it works effectively on SSPs of varying size and difficulty.

Acknowledgements

We gratefully acknowledge the insightful comments by the reviewers. This research was funded in part by the National Science and Engineering Research Council (NSERC), iCore and the Alberta Ingenuity Fund.

References

Asadi, M., & Huber, M. (2004). State space reduction for hierarchical reinforcement learning. In *FLAIRS*, pp. 509 – 514.

Asadi, M., & Huber, M. (2007). Effective control knowledge transfer through learning skill and representation hierarchies. In *IJCAI*, pp. 2054 – 2059.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

BioWare Corp. (1998). Baldur's Gate. November 30, 1998.

Blizzard Entertainment (2002). Warcraft III: Reign of Chaos. July 3, 2002.

Bulitko, V., Björnsson, Y., Luštrek, M., Schaeffer, J., & Sigmundarson, S. (2007). Dynamic Control in Path-Planning with Real-Time Heuristic Search. In *ICAPS*, pp. 49–56.

Bulitko, V., Luštrek, M., Schaeffer, J., Björnsson, Y., & Sigmundarson, S. (2008). Dynamic Control in Real-Time Heuristic Search. *JAIR*. In press.

Bulitko, V., Sturtevant, N., Lu, J., & Yau, T. (2007). Graph Abstraction in Real-time Heuristic Search. *JAIR*, *30*, 51–100.

Dean, T., Givan, R., & Leach, S. (1997). Model reduction techniques for computing approximately optimal solutions for Markov decision processes. In *UAI*, pp. 124–131.

Hauskrecht, M., Meuleau, N., Kaelbling, L. P., Dean, T., & Boutilier, C. (1998). Hierarchical solution of Markov decision processes using macro-actions. In *UAI*, pp. 220 – 229.

Isaza, A., Szepesvári, C., Bulitko, V., & Greiner, R. (2008). Speeding up planning in Markov decision processes via automatically constructed abstraction. Tech. rep., Computing Science, U. Alberta. `http://www.cs.ualberta.ca/~szepesva/RESEARCH/PRMDP`.

Kim, K., & Dean, T. (2003). Solving factored MDPs using non-homogeneous partitions. *Artificial Intelligence*, *147*, 225–251.

McMahan, H. B., & Gordon, G. J. (2005). Fast exact planning in Markov decision processes. In *ICAPS*, pp. 151–160.

Sturtevant, N. (2007). Memory-efficient abstractions for pathfinding. In *AIIDE*, pp. 31–36.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*(1–2), 181–211.

Bayesian Out-Trees

Tony Jebara
Columbia University
New York, NY 10027
jebara@cs.columbia.edu

Abstract

A Bayesian treatment of latent directed graph structure for non-*iid* data is provided where each child datum is sampled with a directed conditional dependence on a single unknown parent datum. The latent graph structure is assumed to lie in the family of directed out-tree graphs which leads to efficient Bayesian inference. The latent likelihood of the data and its gradients are computable in closed form via Tutte's directed matrix tree theorem using determinants and inverses of the out-Laplacian. This novel likelihood subsumes *iid* likelihood, is exchangeable and yields efficient unsupervised and semi-supervised learning algorithms. In addition to handling taxonomy and phylogenetic datasets the out-tree assumption performs surprisingly well as a semi-parametric density estimator on standard *iid* datasets. Experiments with unsupervised and semi-supervised learning are shown on various UCI and taxonomy datasets.

1 INTRODUCTION

Many machine learning methods use graph connectivity structure to constrain the dependencies between random variables or between the samples in a dataset. If the graph structure is latent, Bayesian inference or heuristics are used to recover it. This article explores a distribution over a family of directed graphs known as out-trees where Bayesian inference remains efficient. Furthermore, the directed graph connectivity across samples is helpful not only for structured datasets but *iid* datasets as well.

Graph structure is useful to constrain dependencies between random variables (Pearl, 1988; Meila & Jaakkola, 2006), dependencies across samples [1] in a dataset (Roweis & Saul, 2000; Carreira-Perpinan & Zemel, 2004) or even a heterogeneous combination of the two. It may be acceptable to heuristically choose a single graph structure for some problems (Roweis & Saul, 2000) but, in many settings, a Bayesian treatment of latent graph structure can be more precise (Friedman, 1998; Friedman & Koller, 2003; Kemp et al., 2003; Neal, 2003). Tree structures are a particularly efficient family of subgraphs and are relevant in many real-world non-*iid* datasets spanning dynamics, genetics, biology, decision-making, disease transmission and natural language processing (Leitner et al., 1996; Helmbold & Schapire, 1997; Willems et al., 1995; Mau et al., 1999; Koo et al., 2007). Bayesian inference over undirected trees is efficient (Meila & Jaakkola, 2006) however, graph families beyond undirected trees may require approximate inference. For example, recovering an optimal graph is NP-hard for graphs in families with more than 1 parent per node and requires approximation methods like MCMC sampling which may have slow mixing times (Friedman & Koller, 2003).

This article uses graphs primarily to constrain dependency across exchangeable samples in a dataset. We will assume a latent graph structure was responsible for generating the data and assume it lies in the family of directed out-trees. Like undirected trees, this family of directed graphs also benefits from efficient Bayesian inference algorithms. However, the directed aspect of out-trees is not only beneficial for non-*iid* structured datasets like taxonomy trees it also (surprisingly) improves density estimation for *iid* datasets. We conjecture that the directed tree graph structure assumption acts as a flexible semiparametric estimator that overcomes mismatch between a parametric model and an otherwise *iid* dataset.

[1] In manifold learning, dependencies across samples in a dataset are often constrained using k-nearest neighbor and/or maximum weight spanning tree subgraphs.

This paper is organized as follows. Section 2 describes how out-trees may emerge sequentially in nature and then presents a computationally convenient generative model for them. Section 3 describes Bayesian inference under the latent out-tree assumption and introduces Tutte's directed matrix tree theorem for efficient inference. Section 4 describes an unsupervised maximum likelihood approach to refining the parameters for the out-tree model. Section 5 describes a semi-supervised approach where the out-tree model can be used to transfer inductive bias from inputs to labels. A variational Bayesian setup for integrating over both structure and parameters is described in Section 6. Experiments with unsupervised learning and semisupervised learning are shown in Section 9 and followed by a brief discussion.

2 THE GENERATIVE MODEL

Consider a real-world example of a directed out-tree: the genealogical dataset of neuroscience PhD graduates[2]. This is a population dataset containing $t = 1, \ldots, T$ input samples. Each sample or individual t in this dataset is a node which has a single parent $\pi(t)$, the main doctoral advisor for student t. Also, each node t has a corresponding attribute vector X_t associated with it which describes the student (dissertation topic area, year of graduation, institute, etc.). One realistic way such an out tree is generated is in a sequential or temporal manner. A root node labeled $t = 1$ is sampled with a corresponding vector X_1 of attributes. Knowing the value X_1 for the root, a number of its children nodes are sampled with attributes that depend on the current settings of the parent, X_1. These attribute vectors are drawn from a conditional asymmetric distribution $p(X|X_1)$. This is because an advisor is likely to generate students with related dissertation topics, within a finite number of years of his graduation, and so on. This conditional mimics the process of *mutation* in phylogeny. The children then become parents themselves and go on to generate further descendants by sampling again from the conditional distribution $p(X_t|X_{\pi(t)})$. Assume each attribute vector X_t is a 3D Euclidean vector describing the PhD graduate. One choice for the conditional distribution or mutation is the conditioned Gaussian $\mathcal{N}(X_t|\Sigma_{c|\pi}X_{\pi(t)}, I)$ with a fixed correlation parameter $\Sigma_{c|\pi}$ and identity variance. Figure 1 shows some synthetic out-trees with 3D attribute vectors generated by this conditional.

While the above sequential method for generating real-world data is interesting, computational considerations will encourage us to follow a slightly different generative modeling assumption. The data in Figure 1 was actually generated via a non-sequential recipe as follows. We assume an integer T is given that indicates the total number of nodes. Then, from a prior distribution over trees $p(\mathcal{T})$, an undirected tree $\mathcal{T}$ is chosen to connect the T nodes. Then, we choose a root node from the set of nodes $1, \ldots, T$. We then obtain an out-tree by choosing all edges to point away from the root. Next, the attributes of the root are sampled from a marginal distribution $p(X_r)$. Then, traversing from parent to child along the out-tree, we sample the child's attribute vector X_t according to a conditional (mutation) distribution that depends on its parent $X_{\pi(t)}$ denoted by $p(X_t|X_{\pi(t)})$.

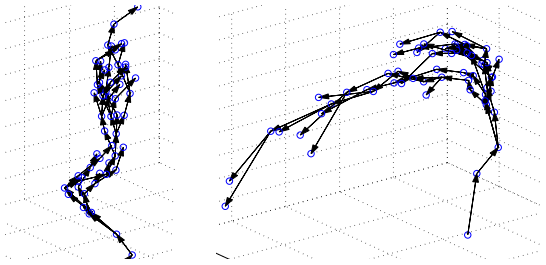


Figure 1: Sampled out-trees using $\mathcal{N}(X_t|\Sigma_{c|\pi}X_{\pi(t)}, I)$.

As is often the case in learning, we assume that some aspects of this generative process are hidden and must be recovered via inference. For instance, we will assume that the tree structure $\mathcal{T}$, the choice of the root and so on are not available to the learner. Instead, as in many real-world datasets, we can only access the node attributes $X_1, \ldots, X_T$ given in some arbitrary ordering. One challenge is to recover the marginal $p(X_r)$ and conditional distributions $p(X_t|X_{\pi(t)})$ from the data. Another challenge is to recover information about the lost connectivity structure $\mathcal{T}$. Another challenge is to recover missing information in some of the attribute vectors, i.e. hidden elements in $X_1, \ldots, X_T$. This article presents efficient Bayesian inference approaches to these problems.

3 EFFICIENT DISTRIBUTIONS OVER OUT-TREES

We are given a training dataset containing input samples X_t for $t = 1 \ldots T$ in some arbitrary order. One quantity to evaluate or manipulate is the likelihood of the dataset $p(X_1, \ldots, X_T|\theta, T)$ given some model. A popular method to recover a model of the dataset is to find the model that maximizes the likelihood score. An additional standard assumption most unsupervised methods make is that the dataset is composed of independently identically distributed samples. In other words, $p(X_1, \ldots, X_T|\theta, T) = \prod_{t=1}^{T} p(X_t|\theta)$. This *iid* assumption can often be inappropriate for real

[2]Available online via http://neurotree.org.

datasets. What is a more minimal set of assumptions on the likelihood function we can make? Likelihoods should be non-negative and sum to unity if we integrate over all $X_1, \ldots, X_T$. In addition, since the data arrives in an arbitrary order, a likelihood should be invariant to permutation of the arguments $\{X_1, \ldots, X_T\}$ for any given finite dataset size T. This property is called finite exchangeability. It is less strict than infinite exchangeability which is in turn less strict than *iid* sampling. [3] The next section derives a likelihood that satisfies these properties yet generalizes the *iid* setting by assuming data was sampled according to an out-tree data structure. The generative model assumes that we first form a complete tree with T nodes (the number T is known a priori) and then children are sampled from their parents using conditional distributions according to the out-tree.

More formally, define an out-tree as an acyclic graph $\mathcal{T}$ with a set of T vertices $\mathcal{X} = X_1, \ldots, X_T$ and directed edges such that each node X_t has at one parent node $X_{\pi(t)}$ and the root has no parents. Note, here we abuse notation and take X_t to refer to the node corresponding to the t'th sample as well as its attribute vector interchangeably. Rooted out-trees are trees with directed edges pointing away from a well-defined root. For instance, $X_1 \leftarrow X_2 \leftarrow X_3$ is an out-tree rooted at X_3. Conversely, rooted in-trees have all directed edges from other nodes point towards the root. The previous 3-chain example is thus also an in-tree rooted at node X_1. Many directed trees are neither in-trees nor out-trees. For instance, the tree $X_1 \rightarrow X_2 \leftarrow X_3 \rightarrow X_4$ is a valid directed tree but neither a rooted in-tree nor a rooted out-tree. For each choice of a root, the set of rooted out-trees forms a disjoint set of T^{T-2} directed trees. Therefore, there are T^{T-1} out-trees for T nodes.

If we knew the latent out-tree structure $\mathcal{T}$ that generated our T samples, the likelihood of the data under the generative assumptions of Section 2 would factorize as a product of conditionals of each node given its parent. However, in general, the structure is unknown. Consider treating structure as a random variable and using Bayes' rule to obtain a posterior distribution over out-trees as follows:

$$p(\mathcal{T}|\mathcal{X}) \quad = \quad \frac{p(\mathcal{X}|\mathcal{T})p(\mathcal{T})}{p(\mathcal{X})} \quad = \quad \frac{\prod_{t=1}^{T} p(X_t|X_{\pi(t)})p(\mathcal{T})}{p(X_1, \ldots, X_T)}.$$

A typical assumption is that the prior over out-tree structures is chosen to be uniform yielding $p(\mathcal{T}) = \frac{1}{\text{card}(\mathcal{T})} = \frac{1}{T^{T-1}}$. This is merely a normalized constant distribution over all possible out-trees. We rewrite the posterior over out-trees as follows:

$$p(\mathcal{T}|\mathcal{X}) \quad = \quad \frac{p(\mathcal{X}|\mathcal{T})}{p(\mathcal{X})T^{T-1}} \quad = \quad \frac{1}{Z}\prod_{t=1}^{T} p(X_t|X_{\pi(t)}). \quad (1)$$

where we have defined the *partition function* Z that ensures that the likelihood term sums to unity over all possible out-trees:

$$Z \quad = \quad p(\mathcal{X})T^{T-1} \quad = \quad \sum_{\mathcal{T}\in\Gamma}\prod_{t=1}^{T} p(X_t|X_{\pi(t)}).$$

Here, $\mathcal{T}$ enumerates over the set of all out-trees, Γ. This is an unwieldy computation since there are T^{T-1} possible out-trees connecting T observation vertices. Instead, we consider breaking up the summation into all possible choices of the root of the out-tree $r = 1 \ldots T$ and a summation over the subset Γ_r of all T^{T-2} out-trees rooted at node r. It is straightforward to show that all subsets of out-trees with different roots are distinct, in other words $\Gamma_i \cap \Gamma_j = \{\}$ if $i \neq j$. Furthermore, their union forms the set of all out-trees $\Gamma = \cup_{j=1}^{T}\Gamma_j$. Thus, the partition function Z is given by the following sum:

$$\begin{aligned} Z \quad &= \quad \sum_{r=1}^{T}\sum_{\mathcal{T}_r\in\Gamma_r}\prod_{t=1}^{T} p(X_t|X_{\pi(t)}) \\ &= \quad \sum_{r=1}^{T} p(X_r)\sum_{\mathcal{T}_r\in\Gamma_r}\prod_{t\neq r}^{T} p(X_t|X_{\pi(t)}) \end{aligned}$$

where we have used the property that the root has no parent node. To efficiently recover Z we will instead recover the individual components of the above sum over r:

$$Z_r \quad = \quad \sum_{\mathcal{T}_r\in\Gamma_r}\prod_{t\neq r}^{T} p(X_t|X_{\pi(t)})$$

by making an appeal to the directed variant of Kirchoff's *Matrix Tree Theorem*, namely Tutte's *Directed Matrix Tree Theorem* (West, 1996). The directed matrix tree theorem does not quite sum over all directed trees. It sums over a *subset*: rooted out-trees. To apply Tutte's theorem we compute an asymmetric β weight matrix of size $T \times T$ populated by all pairwise conditional probabilities $\beta_{uv} = p(X_u|X_v)$. Note that we will assume $\beta_{vv} = 0$ since there are no edges between a node and itself. The matrix β allows us to rewrite Z_r as a product of edges in the tree instead of a product of nodes:

$$Z_r \quad = \quad \sum_{\mathcal{T}_r\in\Gamma_r}\prod_{uv\in\mathcal{T}_r}\beta_{uv}.$$

[3] Another typical assumption most likelihoods require (which is not strictly necessary) is consistency. For instance, a likelihood should produce consistent marginals, i.e. $\sum_{X_T} p(X_1, \ldots, X_T|\theta, T) \neq p(X_1, \ldots, X_{T-1}|\theta, T-1)$. In this article, because of the explicit a priori dependence on the number of samples T, such consistency statements will not be pursued.

The out-tree Laplacian matrix Q is then obtained as follows:

$$Q \;=\; diag(\beta\vec{1}) - \beta.$$

Here, take $\vec{1}$ to be the ones column vector and note that the diag($\vec{v}$) operator gives a diagonal matrix with $\vec{v}$ on its diagonal. Note that this *out Laplacian* is not symmetric since β is not symmetric. Similarly, the *in Laplacian* is given by $Q_{in} = diag(\vec{1}\beta) - \beta$. The directed matrix tree theorem asserts that the number (or weight) of out-trees rooted at node r is Z_r and is given by the matrix cofactor $[Q]_r$ obtained by deleting the r'th row and r'th column of the matrix Q. The precise formula is:

$$Z_r \;=\; |[Q]_r| \;=\; |[\mathrm{diag}(\beta\mathbf{1}) - \beta]_r|\,.$$

Reinserting this formula into the above gives the total partition function as:

$$Z \;=\; \sum_{r=1}^{T} p(X_r) Z_r \;=\; \sum_{r=1}^{T} p(X_r) |[\mathrm{diag}(\beta\mathbf{1}) - \beta]_r|$$

which is now efficient to evaluate. Interestingly, Z is the sum of determinants of the minors of the Laplacian. If β is symmetric, all terms in the summation above are identical and we need to only work with a single determinant of the $T \times T$ matrix. A symmetric β would emerge, for example, if we chose symmetric conditional distributions $p(X_u|X_v) = p(X_v|X_u)$. In addition, it is known that the log determinant of a symmetric Laplacian matrix is a concave function of the edge-weights (Jakobson & Rivin, 2002). In the asymmetric case, however, the log-partition function does not preserve concavity.

A naive implementation recovers Z in $\mathcal{O}(T^4)$ however it is possible in cubic time as in (Jebara & Long, 2005; Koo et al., 2007) via straightforward applications of Woodbury's formula. This is done by creating a matrix of size $(T+1) \times (T+1)$ called $\hat{Q}$:

$$\hat{Q} \;=\; \begin{bmatrix} 1 & \vec{p}^{\,T} \\ -\vec{p} & Q \end{bmatrix}$$

where $\vec{p}$ is a unit-normalized vector of length T whose entries are proportional to the root probabilities:

$$\vec{p}(r) \;=\; \frac{p(X_r)}{\sum_{r=1}^{T} p(X_r)}.$$

The partition function can then be computed by a single evaluation of the matrix determinant as $Z = (\sum_{r=1}^{T} p(X_r))|\hat{Q}|$ which is $\mathcal{O}((T+1)^3)$ although faster methods are also possible (Kaltofen & Villard, 2004). This is an improvement over the summation of smaller determinants which required $\mathcal{O}(T^4)$. In addition, for numerical reasons, we use the logarithm of the partition function. This is recovered via the trace of the matrix logarithm (or the sum of the log-singular values after an SVD) as:

$$\ln Z \;=\; \ln(\sum_{r=1}^{T} p(X_r)) + \mathrm{tr}\left(\ln \begin{bmatrix} 1 & \vec{p}^{\,T} \\ -\vec{p} & Q \end{bmatrix}\right) \quad (2)$$

which takes $\mathcal{O}((T+1)^3)$ time. This computation is more efficient than enumerating over all T^{T-1} out-trees as in the normalized posterior of Equation 1 and makes it possible to consider datasets beyond a thousand points. To scale further, a wide set of approximate methods for large matrices can be leveraged including Nystrom methods (Williams & Seeger, 2001; Drineas & Mahoney, 2005) and column sampling. These methods will be investigated in future work but were not necessary for initial experiments.

4 MAXIMUM *tdid* LIKELIHOOD

An interesting property of the partition function Z is that it forms a finitely exchangeable *tdid* or tree dependent identically distributed likelihood. The likelihood of the data is $p(\mathcal{X}) = ZT^{1-T}$ or more explicitly:

$$p(X_1, \ldots, X_T) \;=\; \frac{1}{T^{T-1}} \sum_{r=1}^{T} p(X_r)\, |[\mathrm{diag}(\beta\mathbf{1}) - \beta]_r| \quad (3)$$

which degenerates into the *iid* likelihood when the conditional dependence between parent and child nodes is extinguished.

Theorem 1 *If the conditional dependence of a child node given a parent node degenerates into the marginal $p(X_t|X_{\pi(t)}) \to p(X_t)$ the* tdid *likelihood simplifies into the* iid *likelihood.*

Proof 1 *Work backwards by writing the* tdid *likelihood in terms of a product over nodes:*

$$p(X_1, \ldots, X_T) = \frac{1}{T^{T-1}} \sum_{r=1}^{T} p(X_r) \sum_{\mathcal{T}_r \in \Gamma_r} \prod_{t \neq r}^{T} p(X_t|X_{\pi(t)}).$$

Removing the dependence on the parent produces:

$$p(X_1, \ldots, X_T) = \frac{1}{T^{T-1}} \sum_{r=1}^{T} \sum_{\mathcal{T}_r \in \Gamma_r} \prod_{t=1}^{T} p(X_t)$$

which then simplifies into the iid *likelihood:*

$$p(X_1, \ldots, X_T) \;=\; \frac{\sum_{r=1}^{T} T^{T-2}}{T^{T-1}} \prod_{t=1}^{T} p(X_t) \;=\; \prod_{t=1}^{T} p(X_t).$$

Thus a generalization of *iid* likelihood emerges by integrating over a latent out-tree sampling structure. One natural way of performing unsupervised learning is to maximize this *tdid* likelihood to recover, for instance, a good setting of the parameters θ that govern the conditional distribution of child given parent. Equation 3 acts as a novel maximum likelihood estimator. We rewrite the *tdid* likelihood to make the dependence on the conditional distribution's parameters θ more explicit:

$$p(X_1, \ldots, X_T|\theta) \quad = \quad \frac{Z(\theta)}{T^{T-1}}.$$

This likelihood satisfies certain desiderata outlined earlier. First, it is invariant to reordering of $\{X_1, \ldots, X_T\}$ and therefore is finitely exchangeable. Furthermore, it is easy to verify that $p(\mathcal{X}) \geq 0$ and sums to unity when integrated over all possible $X_1, \ldots, X_T$.

We next consider maximum likelihood unsupervised learning where we find a θ that produces a large $p(\mathcal{X}|\theta)$. We maximize the log *tdid* likelihood using gradient ascent on the parameters. The gradient for any scalar parameter θ_i is given by the chain rule applied to Equation 2:

$$\frac{\partial \ln Z}{\partial \theta_i} \quad = \quad \frac{1}{\sum_{r=1}^{T} p(X_r)} \sum_{r=1}^{T} \frac{\partial p(X_r)}{\partial \theta_i} + \mathrm{tr}\left(\hat{Q}^{-1} \frac{\partial \hat{Q}}{\partial \theta_i}\right)$$

The main computational requirement for evaluating the gradient is the $\mathcal{O}((T+1)^3)$ matrix inversion. However, the computation of the gradient can easily be approximated for further efficiency.

Given an initial guess for θ, it is possible to follow the gradient or perform line-search. Line search is convenient since evaluating determinants is faster than matrix inversion. Note that maximum likelihood with incomplete information is being performed since we never require anything more than the $\{X_1, \ldots, X_T\}$ population data (there is no additional information about the tree structure).

While any exponential family distribution could be used to specify the marginal and conditional distributions, we focus on the Gaussian case. The following marginal-conditional decomposition of its parameters holds $\theta = \{\mu_c, \mu_\pi, \Sigma_{c|\pi}, \Sigma_{cc}, \Sigma_{\pi\pi}\}$. These are two vectors in $\mathbb{R}^D$ and three matrices in $\mathbb{R}^{D \times D}$. We further assume matrices Σ_{cc} and $\Sigma_{\pi\pi}$ are positive definite. This gives the following Gaussian probabilities for the root nodes:

$$p(X_r|\theta) \quad = \quad \mathcal{N}(X_r|\mu_\pi, \Sigma_{\pi\pi}),$$

and the following conditional Gaussian probability of child given parent:

$$p(X_t|X_{\pi(t)}, \theta) \quad = \quad \mathcal{N}(X_t|\Sigma_{c|\pi} X_{\pi(t)} + \mu_c, \Sigma_{cc}).$$

If we are given a parameter setting and if all X_t variables are observed, it is straightforward to apply these formulae. The resulting probabilities are inserted into the out Laplacian which efficiently recovers the likelihood value or the gradients for unsupervised learning.

Given the gradient and the likelihood evaluation, we can now readily maximize the *tdid* likelihood. However since it is not concave in the exponential family case except when *tdid* degenerates into *iid*, we prefer to initialize with the *iid* solution. For example, in the Gaussian case, a reasonable initialization for θ is to learn the model under *iid* assumptions for the seed model and then perform maximum *tdid* likelihood thereafter.

Once we have learned a model θ^* from training data $\mathcal{X}$, we evaluate the test likelihood on new data $\tilde{\mathcal{X}} = \{\tilde{X}_1, \ldots, \tilde{X}_U\}$ according to:

$$p(\tilde{\mathcal{X}}|\mathcal{X}, \theta^*) = \frac{p(\tilde{\mathcal{X}}, \mathcal{X}|\theta^*)}{p(\mathcal{X}|\theta^*)} = \frac{Z_{\mathcal{X}\cup\tilde{\mathcal{X}}}(\theta^*)}{Z_{\mathcal{X}}(\theta^*)} \frac{T^{(T-1)}}{(T+U)^{(T+U-1)}}.$$

It is straightforward to show that this quantity still integrates to one when we integrate over $\tilde{\mathcal{X}}$. This simply involves computing the partition function for the test data aggregated with the training data $Z_{\mathcal{X}\cup\tilde{\mathcal{X}}}$ relative to the partition function for the training data alone $Z_{\mathcal{X}}$. Both these quantities involve the determinant formula we outlined. Contrast the above test likelihood score to the traditional test likelihood score produced by an *iid* model which simplifies due to factoring:

$$p_{iid}(\tilde{\mathcal{X}}|\mathcal{X}, \theta^*) \quad = \quad \frac{\prod_{j=1}^{T} p(X_j|\theta^*) \prod_{i=1}^{U} p(\tilde{X}_i|\theta^*)}{\prod_{j=1}^{T} p(X_j|\theta^*)}.$$

In *iid* each test point data is independent of the training data and all other test points given the model parameters θ^*. Thus, the *tdid* likelihood is semi-parametric since, in addition to depending on parameters θ, there is a non-parametric dependence on other training and test points. In fact, if we set the conditional Gaussians such that $\Sigma_{c|\pi} = I$ and $\mu_{cc} = 0$, *tdid* can mimic non-parametric Parzen estimation.

5 SEMI-SUPERVISED OUT-TREES

Another application of the latent out-tree assumption is in semi-supervised learning problems (Kemp et al., 2003) where output labels are given in addition to input samples. Consider a label y_t which is generated from a mutation process over the branches of the tree $\mathcal{T}$ just as the attributes of the node X_t are also mutations from their parents. This mutation process defines a distribution over possible labels. For instance, y_t may indicate if an individual in a population has diabetes and X_t is a vector of anatomical features for the individual. Instead of generating a label that depends

only on the input, we could also consider dependence of a child label on a parent input and a parent label. In other words, for training samples X_t and corresponding labels y_t for $t = 1 \dots T$, we have the following likelihood for a known out-tree $\mathcal{T}$:

$$p(X_1, y_1, \dots, X_T, y_T|\mathcal{T}) = \prod_{t=1}^{T} p(X_t, y_t|X_{\pi(t)}, y_{\pi(t)}). \quad (4)$$

The derivations for the partition function proceed as before but now involve a directed Laplacian built from edge weights using conditionals between both inputs and outputs, i.e. $\beta_{uv} = p(X_u, y_u|X_v, y_v)$. We may make more restrictive factorization assumptions on these conditional relationships, for instance, y_t might depend *only* on X_t. A more interesting assumption is y_t is independent of input data altogether and is only conditionally dependent on its parent's label $y_{\pi(t)}$. In other words, the mutation conditional simplifies into $p(y_t|y_{\pi(t)})p(X_t|X_{\pi(t)})$. In an *iid* classification setting, this radical assumption would make learning impossible since output data is independent from input data. However, in a latent out-tree setting, inputs and outputs are only conditionally independent *given* tree structure. When tree structure is unknown, dependence between inputs and outputs emerges without an explicit relationship between input and output spaces (parametric or otherwise). This is because X and y are sampled given the parent random variable $\mathcal{T}$, i.e. $X \leftarrow \mathcal{T} \rightarrow y$. Therefore, observing the input induces a posterior on $\mathcal{T}$ which subsequently induces a posterior on labels. This is particularly useful when we cannot make explicit assumptions about the parametric relationship between the input and output spaces. In these settings, the latent out-tree may be a good source of inductive bias to couple inputs to outputs especially if the number of labeled outputs is small.

To recover the settings of the unobserved y_t labels, one approach is to maximize likelihood while integrating over $\mathcal{T}$. Assume we have observed the labels for the training points $\mathcal{Y} = \{y_1, \dots, y_T\}$ but not for the test points $\tilde{\mathcal{Y}} = \{\tilde{y}_1, \dots, \tilde{y}_U\}$. To predict labels, we need the conditional posterior over unobserved labels given all other observed data, $p(\tilde{\mathcal{Y}}|\mathcal{X}, \tilde{\mathcal{X}}, \mathcal{Y}, \theta)$ as follows:

$$p(\tilde{\mathcal{Y}}|\mathcal{X}, \tilde{\mathcal{X}}, \mathcal{Y}, \theta) = \frac{p(\mathcal{X}, \tilde{\mathcal{X}}, \mathcal{Y}, \tilde{\mathcal{Y}}|\theta)}{\sum_{\tilde{\mathcal{Y}}} p(\mathcal{X}, \tilde{\mathcal{X}}, \mathcal{Y}, \tilde{\mathcal{Y}}|\theta)}.$$

Instead of maximizing the above conditional latent likelihood, we simply maximize the joint latent likelihood (a common simplification in many learning frameworks) to recover parameters θ and unknown labels:

$$p(\mathcal{X}, \tilde{\mathcal{X}}, \mathcal{Y}, \tilde{\mathcal{Y}}|\theta) = \frac{Z(\tilde{\mathcal{Y}}, \theta)}{(T+U)^{(T+U-1)}}$$

where we have rewritten the partition function in terms of both the unknown $\mathcal{Y}$ and unknown θ which need to be specified to construct the out Laplacian $\hat{Q}$. We initialize both randomly and then maximize the partition function using gradient ascent for θ as in the unsupervised learning case and maximize over unknown labels $\hat{\mathcal{Y}}$ by greedily flipping individual labels to increase the partition function. This iterative hill climbing scheme produces a final set of parameters and labels. While investigating a label flip in the output, it is useful to avoid full matrix inversion and full matrix determinants since each label flip only involves a rank 1 change to the out Laplacian matrix $\hat{Q}$ and thus each step of hill climbing requires no more than quadratic time. This and intermediate caching of results allows efficient prediction of labels.

6 VARIATIONAL BAYES

So far we have considered the latent *tdid* likelihood which involves agnostically integrating over all structures $\mathcal{T}$. However, we have assumed that the parameters θ are given or are recovered by a point estimator such as maximum likelihood. A more thorough Bayesian approach is to consider integrating over *both* parameters θ and structure $\mathcal{T}$ after introducing a prior distribution on both. In such a setting, the nonparametric density estimator becomes reminiscent of other nonparameteric Bayesian methods such as Dirichlet processes (Teh et al., 2004; Neal, 2003; Ferguson, 1973) and infinite mixture models (Rasmussen, 1999; Beal et al., 2002). The joint integration over parameters and structure recovers the evidence of the data $p(\mathcal{X})$ or equivalently the log-evidence $\mathcal{E} = \ln p(\mathcal{X})$ (Friedman & Koller, 2003). Consider first splitting the parameters θ into those adjusting the distribution over root θ_m and those adjusting the distribution of child given parent θ_c. We assume the root $p(X_r|\theta_m)$ and conditional distributions $p(X_t|X_{\pi(t)}, \theta_c)$ are in the exponential family and the priors on their parameters $p(\theta) = p(\theta_m)p_c(\theta_c)$ are conjugate. Integrating with a uniform structure prior $p(\mathcal{T}) = \frac{1}{T^{T-1}}$ and making the natural assumption that the prior over structure and parameters factorizes yields:

$$\mathcal{E} = \ln \int_\theta \sum_{r=1}^{T} p(X_r|\theta_m) \sum_{\mathcal{T}_r} \prod_{t \neq r}^{T} p(X_t|X_{\pi(t)}, \theta_c) p(\mathcal{T}) p(\theta)$$

which unfortunately is an intractable quantity. We instead consider a lower bound on the evidence. This is done by introducing variational distributions, for instance, the distribution $q(r)$ over choices for the root

and applying Jensen.

$$\begin{aligned}\mathcal{E} &\geq \sum_{r=1}^{T} q(r) \ln \int_{\theta} p(X_r|\theta_m) \sum_{\mathcal{T}_r} \prod_{t \neq r}^{T} p(X_t|X_{\pi(t)}, \theta_c) p(\theta) \\ &\quad + H(q) - (T-1) \ln T\end{aligned}$$

We also introduce variational distributions over $r = 1, \ldots, T$ out-trees each rooted at node r which we denote by $q_r(\mathcal{T}_r)$ and a variational distribution over the parameters $q_c(\theta_c)$. Re-applying Jensen's inequality produces:

$$\begin{aligned}\mathcal{E} &\geq \sum_{r} q(r) \ln \int_{\theta_m} p(X_r|\theta_m) p(\theta_m) - (T-1) \ln T \\ &\quad + \sum_{r, \mathcal{T}_r} q(r) q_r(\mathcal{T}_r) \sum_{t \neq r}^{T} \int_{\theta_c} q_c(\theta_c) \ln p(X_t|X_{\pi(t)}, \theta_c) \\ &\quad + H(q) + \sum_{r} q(r) H(q_r) - KL(q_c \| p_c).\end{aligned}$$

Above, H denotes the Shannon entropy and KL denotes the Kullback-Leibler divergence. Update rules for each variational distribution iteratively maximize the lower bound by taking derivatives and setting to zero. We update the density over out-trees rooted at node r via:

$$q_r(\mathcal{T}_r) = \frac{1}{Z_r} \prod_{t \neq r}^{T} e^{\int_{\theta_c} q_c(\theta_c) \ln p(X_t|X_{\pi(t)}, \theta_c)}.$$

As in Section 3 this can be rewritten as a product over edges in the out-tree $\mathcal{T}_r$ and summarized simply by a $T \times T$ matrix β whose off diagonal entries are $\beta_{uv} = \int_{\theta_c} q_c(\theta_c) \ln p(X_u|X_v, \theta_c)$. For exponential family $p(X_u|X_v, \theta_c)$, such integrals are easy to solve (Box & Tiao, 1992). Each Z_r is also straightforward to recover using Tutte's theorem. The update for the $q(r)$ distribution is:

$$\begin{aligned}q(r) &\propto e^{H(q_r)} \int_{\theta_m} p(X_r|\theta_m) p(\theta_m) \\ &\quad \times e^{\sum_{\mathcal{T}_r} q_r(\mathcal{T}_r) \sum_{t \neq r}^{T} \int_{\theta_c} q_c(\theta_c) \ln p(X_t|X_{\pi(t)}, \theta_c)}\end{aligned}$$

where the entropy $H(q_r)$ and the expectation over $q_r(\mathcal{T}_r)$ are efficient to compute from the β matrix (Meila & Jaakkola, 2006). Furthermore, the integrals $\int_{\theta_m} p(X_r|\theta_m) p(\theta_m)$ are known for exponential families. We update the distribution over parameters via:

$$\begin{aligned}q_c(\theta_c) &\propto p(\theta_c) e^{\sum_{r, \mathcal{T}_r} q(r) q_r(\mathcal{T}_r) \sum_{t \neq r}^{T} \ln p(X_t|X_{\pi(t)}, \theta_c)} \\ &= p(\theta_c) \prod_{u \neq v} p(X_u|X_v, \theta_c)^{\sum_{r, \mathcal{T}_r} q(r) q_r(\mathcal{T}_r) \delta(uv \in \mathcal{T}_r)}.\end{aligned}$$

This is simply the prior times a product over all pairs of data-points likelihoods with different weights $q_c(\theta_c) \propto p(\theta_c) \prod_{u \neq v} p(X_u|X_v, \theta_c)^{W_{uv}}$. These weights are recovered easily from the current β matrix.

A variational Bayesian treatment is possible over joint out-tree structure and parameters. The method allows us to refine a lower bound on evidence and permits full (nonparametric) Bayesian estimation with out-trees.

7 STATIONARY MUTATION

It is helpful to distinguish some important differences between the Bayesian averaging over out-trees here and the Bayesian inference of tree belief networks presented in (Jaakkola et al., 1999; Meila & Jaakkola, 2006). While elegantly providing a tractable computation of the Bayesian inference, (Meila & Jaakkola, 2006) makes no requirement on the stationarity of the conditional distribution $p(X_t|X_{\pi(t)}, \theta_c)$ which is a key distinguishing component of the out-tree framework. In other words, in (Jaakkola et al., 1999; Meila & Jaakkola, 2006) each conditional has its own parameter θ_t and the samples are drawn from custom conditionals $p(X_t|X_{\pi(t)}, \theta_t)$. If one assumes *decomposable* priors, the Bayesian evidence and Bayesian inference jointly over parameter and structure is elegantly tractable. However, it explores distinct θ_t for all conditionals. This article introduces the constraint that $\theta_t = \theta_{t'} = \theta_c$ for all $t = 1, \ldots, T$ except for the root node. This constraint greatly restricts the model and assumes that the mutation distribution is stationary across all samples. In other words the parameters of the conditional are fixed. This is a key difference and permits us to recover the *iid* setting as a special case when the conditional dependence is extinguished. The unrestricted Bayesian inference method in (Meila & Jaakkola, 2006) can be seen as a step in the variational Bayesian procedure since the update rule for the distribution $q_c(\theta_c)$ collapses the individual conditionals into a single θ_c model.

8 HILBERT GAUSSIANS

In addition to the Gaussian, many exponential family choices for the marginal distribution over root attributes $p(X_r|\theta_m)$ and for the conditional $p(X_u|X_v, \theta_c)$ are possible and computationally convenient. One useful feature of the Gaussian is that it is readily converted into conditional form and leads to a flexible linear relationship between parent and child which is determined primarily by the variable $\Sigma_{c|\pi}$. To go beyond this linear relationship, we may use a mapping on the features or attributes of the parent node which in no way changes the normalization properties of the Gaussian. Thus, we may consider first mapping the parent's features into a higher dimensional vector rep-

Dataset (D, T)	Spiral $(3, 534)$	Heart (13,139)	Diabetes (8,268)	Liver (6,200)
Parzen	-5.61e3	-1.94e3	-6.25e3	-3.41e3
GMM-1	-1.36e3	-2.02e4	-2.12e5	-2.53e4
GMM-2	-1.36e3	-3.23e4	-2.85e5	-1.88e4
GMM-3	-1.19e3	-2.50e4	-4.48e5	-2.79e4
GMM-4	-7.98e2	-1.68e4	-2.03e5	-2.62e4
GMM-5	-6.48e2	-3.15e4	-3.40e5	-3.23e4
GMM-∞	-4.86e2	-4.02e2	-8.22e2	-4.56e2
tdid	-3.91e2	-5.29e2	-8.87e2	-4.99e2

Table 1: Gaussian test log-likelihoods using RBF Parzen estimators, EM mixtures of Gaussians, the ∞ Gaussian mixture model, and the *tdid* estimator.

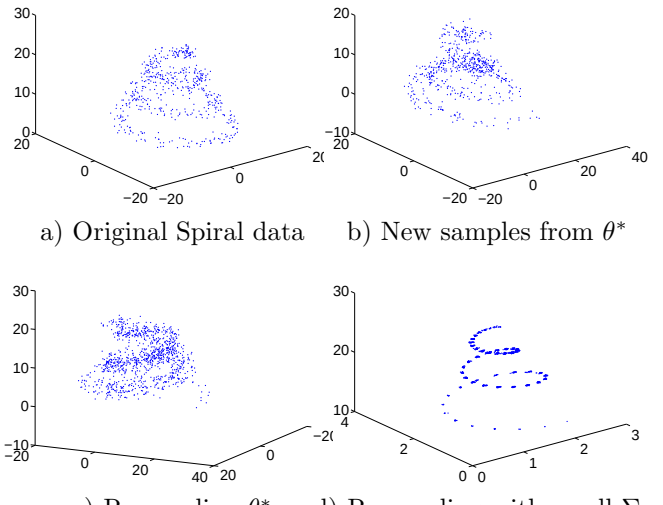
resentation $\phi(X_{\pi(t)})$ of dimensionality $\mathbb{R}^H$ and then learning a matrix $\Sigma_{c|\pi}$ of size $D \times H$. This is a generalized linear conditional model that can capture more complex relationships between the parent and child nodes. In such a setting, the conditional Gaussian relationship need not be represented explicitly in the space of $\phi(X_{\pi(t)})$ but only implicitly in kernelized form over each dimension of the input space:

$$p(X|X_\pi) = \prod_{d=1}^{D} \mathcal{N}\left(X(d) \middle| \sum_{t=1}^{T} \alpha_{t,d} k(X_\pi, X_t) + \mu_d, \sigma_d\right)$$

where the unconditional means μ_d, variances σ_d and weights $\alpha_{t,d}$ are scalars for $t = 1, \ldots, T$ and $d = 1, \ldots, D$, the latter of which indexes the dimensions of the attributes. Furthermore, the function $k(.,.)$ can be any kernel that maps a pair of inputs in the sample space into a scalar measurement of affinity. This gives a general way of extending the linearity assumptions in the conditional model. Instead of making the conditional dependence linear in the values of the parent attributes, we can explore linearity in any *features* of the parent attributes which leads to another source of nonparametric flexibility in the estimator.

9 EXPERIMENTS

To visualize the out-tree model's ability to fit data, we estimated marginal and conditional Gaussian parameters using the latent likelihood for the UCI Spiral dataset in Figure 2(a). Once the $\theta = \{\mu_c, \mu_\pi, \Sigma_{c|\pi}, \Sigma_{cc}, \Sigma_{\pi\pi}\}$ parameters were recovered (a total of 27 scalar degrees of freedom), they were used to generate synthetic datasets of 600 samples in Figures 2(b), (c) and (d). In the last example, the matrix Σ_{cc} was reduced to sample a spiral with less noise. Notice how the datasets can produce slightly different spirals that may have more or fewer turns but still maintain the appropriate overall shape.



a) Original Spiral data b) New samples from θ^*

c) Resampling θ^* d) Resampling with small Σ_{cc}

Figure 2: Spiral data density estimation.

We next show more quantitative unsupervised density estimation experiments on standard UCI datasets where a large test log-likelihood implies a better density estimate. These experiments closely follow the format in (Jebara et al., 2007). Table 1 summarizes the results with various Gaussian models including the marginal-conditional Gaussian model for the *tdid* out-tree approach. On 4 standard datasets (we only use one class for labeled datasets), the test log-likelihood was evaluated after using a variety of density estimators. These estimators include a nonparametric Parzen RBF estimator with a varying scale parameter σ. In addition, a mixture of 1 to 5 Gaussians were fit using Expectation Maximization to maximize *iid* likelihood. Comparisons are also shown with semiparametric density estimators like the infinite mixture of Gaussians (Rasmussen, 1999). Cross-validation was used to choose the σ, and EM local minimum (from ten initializations), for the Parzen and EM approaches respectively. Similarly, cross-validation was used to early-stop the Bayesian out-tree maximum likelihood gradient ascent procedure although this did not have a large effect on performance. Train, cross-validation and test split sizes where 80%, 10% and 10% respectively. The 10 fold averaged test log-likelihoods show that the new method outperforms traditional mixture modeling and Parzen estimators and is comparable to semiparametric methods such as the infinite Gaussian mixture ($iid - \infty$) model (Rasmussen, 1999). Despite the cubic time linear algebra steps for *tdid* estimation, the infinite Gaussian mixture model was the most computationally demanding method.

In a semi-supervised learning problem, we evaluated how well the latent out-tree structure works for clas-

sification and its ability to perform inductive transfer. First, θ is learned from only input samples in an unsupervised manner and the Gaussian parameters $\Sigma_{c|\pi}$ and $\Sigma_{cc} \propto I$ are recovered as above by maximizing the partition function. Then, observed labels are used in the following conditionals to construct out Laplacian:

$$\begin{aligned} p(X_u|X_v) &= \mathcal{N}(X_u|\Sigma_{c|\pi}X_v, \sigma I) \\ p(y_u|y_v) &= \alpha I(y_u = y_v) + (1-\alpha)I(y_u \neq y_v). \end{aligned}$$

Here, the mutation on the outputs is independent of the mutation on the inputs and is simply built from indicator functions with a parameter α which controls the *stickiness* of the label across parent to child. Since only some labels are known, the unknown ones are initialized randomly and improved by maximizing Z. This is done with $\Sigma_{c|\pi}$ and $\Sigma_{cc} \propto I$ locked from their unsupervised values. The unknown discrete labels are greedily explored to further increase the partition function. For comparison, support vector machines were trained on the labeled X and y data.

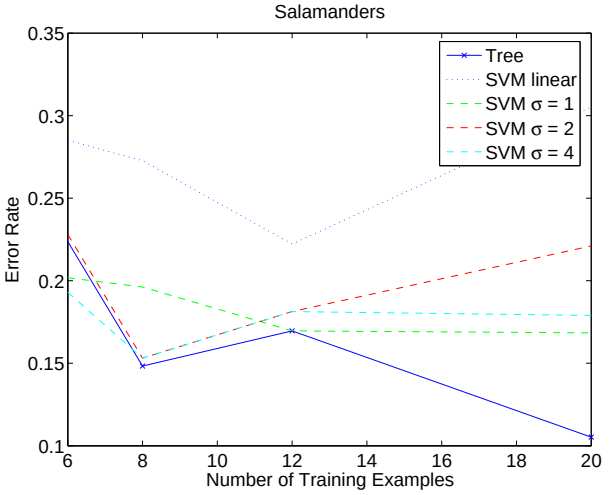


Figure 3: Labeling error rates (averaged over tasks) for Out-Trees and SVMs for salamanders taxonomy.

To evaluate the semi-supervised learning method, two taxonomic datasets (Salamanders and Crustaceans) were used as in (Kemp et al., 2003). These are groups of $T = 30$ and $T = 56$ species nodes and $D = 19$ and $D = 74$ attribute dimensions respectively. Each has a number of discrete attributes describing the external anatomy of each species. These datasets do not have class labels. Therefore each attribute was in turn used as a label to be predicted from the input data. Each dataset therefore generates D discrete prediction tasks. To reduce dimensionality and avoid redundancies (some attributes have the same settings as the target predictions), the remaining $D-1$ attributes were converted into 3D coordinates using PCA before being used as inputs (both the SVM and the out-tree method are similarly hindered by the resulting loss of information). The input attributes for each problem is a set of 3D vectors $X_1, \ldots, X_T$ and the targets are the original discrete-valued $y_1, \ldots, y_T$ labels.

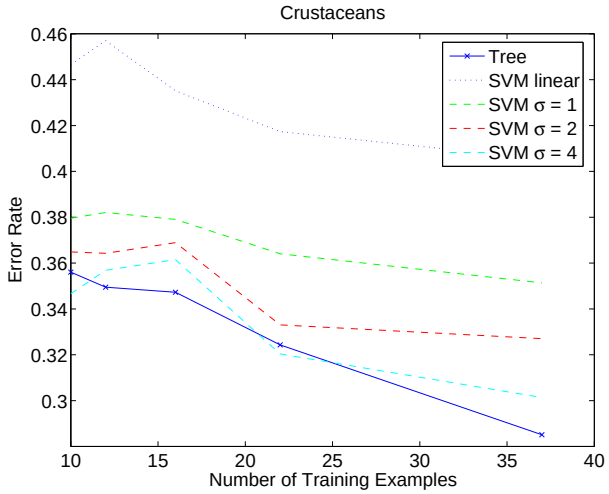


Figure 4: Labeling error rates (averaged over tasks) for Out-Trees and SVMs for crustaceans taxonomy.

Each task was split into training and testing components and the out-tree model was fit and used to find labels. Results were compared with an SVM baseline classifier using different kernel functions. In all experiments for a given number of labeled examples, half the unlabeled examples were used for cross-validation of α for the out-tree model and C for the SVM. The remaining half of the unseen labels were used for testing. Figure 3 shows the average error rate on random folds for all tasks as the number of training examples is varied for salamander species taxonomy. Similarly, Figure 4 shows the crustaceans taxonomy. The out-tree has a statistically significant advantage over both linear and RBF SVMs when classifying data that obeys a directed tree structure.

10 DISCUSSION

This article described a Bayesian treatment of a latent directed out-tree connectivity on non-*iid* data-points. This led to a generative model appropriate for taxonomy and tree data as well as an interesting semi-parametric density estimator for datasets in general. The matrix tree theorem was extended to directed trees and enjoys the same efficient Bayesian inference properties as its undirected counterpart. A novel *tdid* likelihood emerges which permits the recovery of both a marginal density on nodes as well as the conditional of each node given its latent parent. The new likelihood is exchangeable and is a direct generalization of *iid* likelihood. It degenerates into *iid* when conditional

dependencies between children and parents collapse. A variational Bayesian treatment is also possible by integrating over both parameters and out-tree structures jointly. Experiments with unsupervised and semisupervised learning were promising.

Acknowledgments

The author thanks A. Howard, C. Kemp, P. Long, J. Tenenbaum and the anonymous reviewers for their comments and for providing data. This work was supported by ONR Award N000140710507 (Mod No: 07PR04918-00) and NSF Award IIS-0347499.

References

Beal, M., Ghahramani, Z., & Rasmussen, C. (2002). The infinite hidden Markov model. *NIPS*.

Box, G., & Tiao, G. (1992). *Bayesian inference in statistical analysis*. John Wiley & Sons.

Carreira-Perpinan, M., & Zemel, R. (2004). Proximity graphs for clustering and manifold learning. *NIPS*.

Drineas, P., & Mahoney, M. (2005). On the Nystrom method for approximating a gram matrix for improved kernel-based learning. *JMLR*, *6*, 2153–2175.

Ferguson, T. (1973). A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, *1*, 209–230.

Friedman, N. (1998). The Bayesian structural EM algorithm. *UAI*.

Friedman, N., & Koller, D. (2003). Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, *2*, 95–125.

Helmbold, D., & Schapire, R. (1997). Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, *27*, 56–68.

Jaakkola, T., Meila, M., & Jebara, T. (1999). Maximum entropy discrimination. *Neural Information Processing Systems (NIPS)*.

Jakobson, D., & Rivin, I. (2002). Extremal metrics on graphs. *Forum Math*, *14*.

Jebara, T., & Long, P. (2005). *Tree dependent identically distributed learning* (Technical Report CUCS-040-05). Columbia University, Computer Science.

Jebara, T., Song, Y., & Thadani, K. (2007). Density estimation under independent *similarly* distributed sampling assumptions. *NIPS*.

Kaltofen, E., & Villard, G. (2004). Computing the sign or the value of the determinant of an integer matrix, a complexity survey. *J. Comp. Applied Math*, *162*, 133–146.

Kemp, C. amd Griffiths, T., Stromsten, S., & Tenenbaum, J. (2003). Semi-supervised learning with trees. *NIPS*.

Koo, T., Globerson, A., Carreras, X., & Collins, M. (2007). Structured prediction models via the matrix-tree theorem. *EMNLP*.

Leitner, T., Escanilla, D., Franzen, C., Uhlen, M., & Albert, J. (1996). Accurate reconstruction of a known HIV-1 transmission history by phylogenetic tree analysis. *Proceedings of the National Academy of Sciences*, *93*, 10864–9.

Mau, B., Newton, M., & Larget, B. (1999). Bayesian phylogenetic inference via Markov chain Monte Carlo methods. *Biometrics*, *55*, 1–12.

Meila, M., & Jaakkola, T. (2006). Tractable Bayesian learning of tree belief networks. *Statistics and Computing*, *16*, 77–92.

Neal, R. (2003). Density modeling and clustering using Dirichlet diffusion trees. *Bayesian Statistics*, *7*, 619–629.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann.

Rasmussen, C. (1999). The infinite Gaussian mixture model. *NIPS*.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*.

Teh, Y., Jordan, M., Beal, M., & Blei, D. (2004). Hierarchical Dirichlet processes. *NIPS*.

West, D. (1996). *Introduction to graph theory*. Prentice Hall.

Willems, F., Shtarkov, Y., & Tjalkens, T. (1995). The context-tree weighting method: basic properties. *IEEE Transactions on Information Theory*, *41*, 653–664.

Williams, C., & Seeger, M. (2001). Using the Nystrom method to speed up kernel machines. *NIPS*.

Feature Selection via Block-Regularized Regression

Seyoung Kim
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Eric Xing
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Identifying co-varying causal elements in very high dimensional feature space with internal structures, e.g., a space with as many as millions of linearly ordered features, as one typically encounters in problems such as whole genome association (WGA) mapping, remains an open problem in statistical learning. We propose a block-regularized regression model for sparse variable selection in a high-dimensional space where the covariates are linearly ordered, and are possibly subject to local statistical linkages (e.g., block structures) due to spacial or temporal proximity of the features. Our goal is to identify a small subset of relevant covariates that are not merely from random positions in the ordering, but grouped as contiguous blocks from large number of ordered covariates. Following a typical linear regression framework between the features and the response, our proposed model employs a sparsity-enforcing Laplacian prior for the regression coefficients, augmented by a 1st-order Markovian process along the feature sequence that "activates" the regression coefficients in a coupled fashion. We describe a sampling-based learning algorithm and demonstrate the performance of our method on simulated and biological data for marker identification under WGA.

1 INTRODUCTION

Recent advances in high-throughput genotyping technology have allowed researchers to generate a high volume of genotype data at a relatively low cost. An association study involves examining genotype data of individuals in a population and phenotype data for the same individuals such as disease status, gene expression, and physiological measurements in order to discover genetic markers that affect the phenotype of the individual possessing a particular variation of the marker. Variations in a single nucleotide called single nucleotide polymorphisms (SNPs) provide a useful set of genetic markers since they are relatively common across the genome. The whole-genome association study has become feasible because of the relatively low cost involved in typing a large number of SNPs. The challenge in this type of study is to identify a small subset of SNPs associated with the phenotype among the full set of SNPs that can be as large as 8 million (the International HapMap Consortium, 2005).



Figure 1: An illustration of linkage disequilibrium in chromosomes in an association study.

A simple single-marker test has been widely used for detecting an association (Stranger et al. 2005, Cheung et al. 2005). Using this approach, one examines the correlation between the given phenotype and frequencies of each polymorphic allele of one SNP marker at a time to compute p-value of the SNP, and finds the SNPs with low p-values to be significant. The single-marker test assumes that SNPs are independent of each other, ignoring an important correlation structure due to the *linkage disequilibrium* present in the sequence of SNPs. In reality, the states of SNPs that are adjacent in the genome can be tightly coupled (i.e., in linkage disequilibrium). This is because when an individual inherites a chromosomal material from each of the parents, a recombination event can break the parental chromosomes into non-random inheritable segment, causing SNPs within the segment to be inherited with high probability, and preventing random combinations of all possible SNP states within the segment. Since the recombination sites are non-uniformly distributed across the genome, recombination events in chromosomes in a population over generations lead to a block structure in SNPs on the

chromosome. As illustrated in Figure 1, each chromosome is a mosaic of ancestor chromosomes, where segments of SNPs of the same color have been inherited from the same ancestor chromosome. The true association SNPs called causal SNPs are indicated as circles in Figure 1. Since a chromosome segment carrying causal alleles can be inherited as a block, we can take advantage of this block structure to increase the power of the study for detecting association by considering a block of linked SNPs jointly rather than a single SNP at a time.

A multi-marker approach takes into account this linkage disequilibrium pattern by testing a short segment of SNP markers called a haplotype for an association (Zailen et al. 2007, Zhang et al. 2002). In this case, a haplotype instead of a single SNP acts as a proxy for untyped causal SNPs. However, they test a haplotype of a fixed length for an association, scanning the genome using a sliding window. Most of these approaches do not explicitly make use of the block structure with possibly varying block lengths in the sequence of SNPs.

In this paper, we propose a model for association mapping that explicitly incorporates the linkage disequilibrium pattern. We focus on continuous valued phenotypes, and base our method on a linear regression model, where the SNPs are predictors and the phenotypes are response variables. The SNPs with large regression coefficients are found to be significant. The number of SNPs involved in a typical association study is very large, and we are interested in extracting a small number of causal SNPs that are grouped into blocks due to linkage disequilibrium. Thus, we can view this problem as 1) identifying relevant covariates when the covariates lie in a high-dimensional space and 2) learning a block structure in those relevant covariates at the same time.

To enforce sparsity in the regression model, we use the Laplacian prior on the regression coefficients, similar to the L_1 penalty in the lasso (Tibshirani 1996). However, the lasso does not provide any mechanism to incorporate the correlation structure in covariates into the model to address the second problem. In this paper, we propose to encode the information of the correlation pattern in the SNPs as a Markov chain, and use this Markov chain as a prior in the regression model. The block boundaries for chromosome regions with a high level of association are determined probabilistically through transition probabilities in the Markov chain. A regression-based association has been used previously (Servin and Stephens 2007), but they did not address the problem of taking into account the block structure in the genome in a high dimensional space to improve the power of the study.

There is a large body of literature on variable selection methods such as the lasso (Tibshirani 1996) and Bayesian variable selection algorithms (George and McCulloch 1993, Ishwaran and Rao 2005, Yuan and Lin, 2005). Most of these works did not consider situations in which the covariates are structured in a certain manner. Nott and Green (2004) used the correlation information in the covariates to improve the convergence of sampling algorithm, but the model itself did not assume any structure in the covariates. In the fused lasso (Tibshirani et al. 2005), covariates were assumed to be ordered, and adjacent regression coefficients tended to be fused to take the same value, encouraging sparsity in the difference between adjacent coefficients. However, it used only the ordering information, and did not take into account the additional information on the structure in covariates such as the linkage disequilibrium pattern in the case of association study. In the group lasso (Yuan and Lin 2005), the group structure in covariates was assumed to be known, whereas in our proposed model we determine the block structure of relevant covariates during learning given prior knowledge on the correlation structure.

The rest of the paper is organized as follows. In Section 2, we describe the proposed model for association mapping and the learning algorithm. In Section 3, we apply the model to simulated data and mouse data, and compare the performance of the proposed model with that of existing methods. In Section 4, we conclude with a brief discussion of future work.

2 GENOME-WIDE ASSOCIATION VIA BLOCK-REGULARIZED REGRESSION

2.1 THE MODEL

The Association Model Let us assume that a set of J SNP markers have been typed for N individuals. The genotype of each SNP in an individual consists of two alleles corresponding to each of a pair of chromosomes in the case of diploid organisms such as humans and mice. For our regression analysis, we construct an $N \times J$ design matrix $\mathbf{X}$, where each element x_{ij} takes values from $\{0, 1, 2\}$ according to the number of minor alleles in the genotype of the jth SNP of the ith individual, assuming a strictly additive genetic effect of the minor allele. Note that in our analysis the J SNP markers are assumed to be ordered in terms of their positions on chromosome. In addition, a measurement of phenotype y_i is available for each individual i, and we let $\mathbf{y}$ be an $N \times 1$ vector of such measurements for the N individuals. In particular, the covariates in $\mathbf{X}$ lie in a high dimensional space with only a small sub-

set of the J SNPs influencing the output $\mathbf{y}$. In this setting, we assume a standard linear regression model as follows:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon, \quad \epsilon \sim N(0, \sigma^2),$$

where $\boldsymbol{\beta}$ is a vector of J regression coefficients $\{\beta_1, \ldots, \beta_J\}$, and the noise ϵ is modeled as having a normal distribution with mean 0 and variance σ^2. Taking a Bayesian approach, we set the prior for σ^2 to Inv-gamma$(\nu_0/2, (\nu_0 s_0^2)/2)$.

We use a prior on $\boldsymbol{\beta}$ that enforces sparsity in the coefficients. We model each regression coefficient β_j as coming from a mixture of two components, one component representing irrelevant covariates (non-causal SNPs) and the other for relevant covariates (causal SNPs). We introduce a random variable c_j that takes values from $\{0, 1\}$ to indicate the mixture component label for β_j. If $c_j = 0$, the jth SNP is non-causal, and β_j is set to 0. If $c_j = 1$, then we model β_j as coming from a Laplacian distribution. The complete probability distribution for β_j given c_j is given as

$$\beta_j | c_j \sim \begin{cases} I(\beta_j = 0) & \text{if } c_j = 0 \\ \frac{1}{2(2\lambda\sigma^2)} \exp\left(-\frac{|\beta_j|}{2\lambda\sigma 2}\right) & \text{if } c_j = 1, \end{cases} \quad (1)$$

where λ is the parameter that controls the amount of sparsity. We use Inv-gamma(α, γ) as a prior distribution on λ, and sample λ from its posterior during learning (Park and Casella 2008).

The Laplacian prior in Equation (1) is the same as the L_1 penalty used in the lasso (Tibshirani 1996), and has been shown to be useful in a more general problem of learning a sparse model in high-dimensional space (Wainwright et al. 2006). In a Bayesian setting, models similar to the one described above have been used in the literature of Bayesian variable selection with various different distributions in Equation (1) (George and McCulloch 1993, Ishwaran and Rao, 2005, Yuan and Lin, 2005). In almost all of these works, c_j's were modeled as coming from a Bernoulli distribution with parameter p, assuming that c_j's are independent of each other. This independence assumption is not appropriate in the case of genetic data since nearby SNP markers are known to be highly correlated due to the linkage disequilibrium. In the next section, we propose to use a Markov chain prior for c_j's that takes advantage of this dependency.

Markov Chain Prior for Block Structure Because of the linkage disequilibrium, individual chromosomes tend to have a block structure in the sequence of genetic markers, and such blocks of genetic markers are shared across individuals in a population. Recombination rate summarizes the degree of correlation between tightly linked SNPs. In a region with a high recombination rate, the previously linked SNPs are likely to be decoupled, resulting in a weak correlation, whereas a segment of tightly linked SNPs is preserved in the absence of recombination during inheritance. Becuase of the linkage disequilibrium structure, considering a block of highly correlated SNPs instead of a single SNP for an association with the phenotype can potentially increase the power of the association study for detecting causal SNPs.

In this section, we propose to use a Markov chain prior on the indicator variable $\mathbf{c}$ to take into account the block structure due to the linkage disequilibrium in SNP markers by incorporating the recombination rate information in the prior. There is a large body of literature on estimating recombination rates given genotype data of unrelated individuals from a population (Fearnhead and Donnelly 2001, Li and Stephens 2003, Sohn and Xing, 2007). Any of these methods can be used to estimate recombination rates as part of a preprocessing step prior to the association analysis through the proposed method.

Given the estimated recombination rates, we assume that the J covariates are ordered in their positions to have a chain structure and that there is an implicit block structure in the chain where the block boundaries are defined stochastically in terms of the distance and recombination rate between each pair of covariates. In this setting, a group of SNPs within a block can be assigned together to be either causal or non-causal.

We model the sequence of indicator variables $\mathbf{c} = \{c_1, \ldots, c_J\}$ as a Markov chain as follows:

$$P(\mathbf{c}) = P(c_1) \prod_{j=2}^{J} P(c_j | c_{j-1}).$$

For $P(c_j|c_{j-1})$, we use a Poisson process model, a model commonly used for recombination process (Li and Stephens 2003),

$$P(c_j | c_{j-1}) = \exp(-d_j \rho_j)\ \delta(c_j, c_{j-1}) + (1 - \exp(-d_j \rho_j))\ \Pi_{c_{j-1}, c_j}, \quad (2)$$

where Π is a transition matrix $\begin{pmatrix} \pi_0 & 1-\pi_0 \\ 1-\pi_1 & \pi_1 \end{pmatrix}$, d_j is the distance between two adjacent SNPs at positions $(j-1)$ and j on chromosome, and ρ_j is the recombination rate for the same interval. The first term on the right-hand side of Equation (2) corresponds to the probability of no recombination events between the $(j-1)$th and the jth SNPs. On the other hand, the second term models a transition in the presence of a recombination event between the two SNPs. At recombination, the model can either transition to the

same state, or to a different state. If the distance d_j between the $(j-1)$th and jth SNPs is small or the recombination rate ρ_j is low, the two SNPs are tightly linked, and it is likely that both SNPs will receive the same assignment for c_{j-1} and c_j. Thus, the c_j's for causal SNPs are set to 1 in a coupled manner, activating the corresponding covariates to take non-zero regression coefficients. We place a prior $\text{Beta}(a_{00}, b_{00})$ on π_0, and $\text{Beta}(a_{10}, b_{10})$ on π_1.

Our model differs from the fused lasso (Tibshirani et al. 2005) in that it encourages adjacent correlated covariates to take on the same assignment of whether they are relevant or not, while allowing each covariate to have its own regression coefficient. In the fused lasso, the adjacent regression coefficients themselves are encouraged to have the same values. In addition, our method directly makes use of the additional information in covariates such as the distance and recombination rate between two SNPs, whereas the fused lasso only uses the ordering information in covariates to fuse coefficients.

2.2 PARAMETER ESTIMATION

Because of the non-differentiability of the function used in Equation (1) for β_j when $c_j = 0$, it is not possible to learn parameters of the model using an EM style algorithm commonly used for hidden Markov models. Instead, we use the Gibbs sampling to learn the parameters $\Theta = \{\boldsymbol{\beta}, \mathbf{c}, \sigma^2, \Pi\}$ of the model. In this section, we derive conditional posterior distributions of the parameters for Gibbs sampling.

For each of the J covariates, we sample β_j and c_j from their joint posterior distribution

$$p(\beta_j, c_j|\boldsymbol{\beta}_{-j}, \mathbf{c}_{-j}, \mathbf{y}, \mathbf{X}, \sigma^2) = p(\beta_j|\boldsymbol{\beta}_{-j}, \mathbf{c}, \mathbf{y}, \mathbf{X}, \sigma^2) P(c_j|\boldsymbol{\beta}_{-j}, \mathbf{c}_{-j}, \mathbf{y}, \mathbf{X}, \sigma^2). \quad (3)$$

We first sample c_j from the marginal distribution, the second term on the right-hand side of Equation (3), after integrating out β_j. Conditional on the sampled c_j, we sample β_j from its conditional posterior, the first term on the right-hand side of Equation (3).

In order to sample c_j, we re-write the second term on the right-hand side of Equation (3) as

$$P(c_j = k|\boldsymbol{\beta}_{-j}, \mathbf{c}_{-j}, \mathbf{y}, \mathbf{X}, \sigma^2) \propto p(\mathbf{y}|\boldsymbol{\beta}_{-j}, c_j = k, \mathbf{X}, \sigma^2) P(c_j = k|c_{j-1}) P(c_{j+1}|c_j = k)$$

Sampling from the above equation requires to compute the marginal likelihood $p(\mathbf{y}|\boldsymbol{\beta}_{-j}, c_j, \mathbf{X}, \sigma^2)$ after integrating out β_j when $c_j = 0$ and $c_j = 1$. When $c_j = 0$, the jth covariate is irrelevant, and we set $\beta_j = 0$. Thus, the marginal likelihood is simply given as

$$p(\mathbf{y}|\boldsymbol{\beta}_{-j}, c_j = 0, \mathbf{X}, \sigma^2) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \exp\left(-\frac{\sum_i (y_i - \mathbf{x}_i\boldsymbol{\beta})^2}{2\sigma^2}\right).$$

When $c_j = 1$, we compute the integral as below.

$$\begin{aligned}
&p(\mathbf{y}|\boldsymbol{\beta}_{-j}, c_j = 1, \mathbf{X}, \sigma^2) \\
&= \int_{-\infty}^{\infty} \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \exp\left(-\frac{\sum_i (y_i - \sum_k x_{ik}\beta_k)^2}{2\sigma^2}\right) \\
&\quad \cdot \frac{1}{2(2\lambda\sigma^2)} \exp\left(-\frac{|\beta_j|}{2\lambda\sigma^2}\right) \mathrm{d}\beta_j \\
&= K \int_{-\infty}^{\infty} \exp\left(-\frac{\sum_i (z_i - x_{ij}\beta_j)^2 + \frac{|\beta_j|}{\lambda}}{2\sigma^2}\right) \mathrm{d}\beta_j \\
&= K \left(\int_{-\infty}^{0} \exp\left(-\frac{\sum_i (z_i - x_{ij}\beta_j)^2 - \frac{\beta_j}{\lambda}}{2\sigma^2}\right) \mathrm{d}\beta_j \right. \\
&\quad \left. + \int_{0}^{\infty} \exp\left(-\frac{\sum_i (z_i - x_{ij}\beta_j)^2 + \frac{\beta_j}{\lambda}}{2\sigma^2}\right) \mathrm{d}\beta_j \right), \qquad (4)
\end{aligned}$$

where $z_i = y_i - \sum_{k/j} x_{ik}\beta_k$, and $K = (1/2\pi\sigma^2)^{\frac{N}{2}}/(2(2\lambda\sigma^2))$.

Let $A_{(-)}$ denote the first integral and $A_{(+)}$ the second integral in Equation (4). Then, using a straightforward algebra, it can be shown that $A_{(-)}$ and $A_{(+)}$ are given as

$$\begin{aligned}
A_{(-)} &= \exp\left(-\left(\sum_i z_i^2 - \frac{(\sum_i z_i x_{ij} + 0.5/\lambda)^2}{\sum_i x_{ij}^2}\right)/(2\sigma^2)\right) \\
&\quad \cdot \sqrt{\frac{2\pi\sigma^2}{\sum_i x_{ij}^2}} \int_{-\infty}^{0} N_{(-)} \, \mathrm{d}\beta_j \\
A_{(+)} &= \exp\left(-\left(\sum_i z_i^2 - \frac{(\sum_i z_i x_{ij} - 0.5/\lambda)^2}{\sum_i x_{ij}^2}\right)/(2\sigma^2)\right) \\
&\quad \cdot \sqrt{\frac{2\pi\sigma^2}{\sum_i x_{ij}^2}} \int_{0}^{\infty} N_{(+)} \, \mathrm{d}\beta_j,
\end{aligned}$$

where

$$\begin{aligned}
N_{(-)} &= N\left(\beta_j | \frac{\sum_i z_i x_{ij} + \frac{1}{2\lambda}}{\sum_i x_{ij}^2}, \sigma^2 (\sum_i x_{ij}^2)^{-1}\right) \\
N_{(+)} &= N\left(\beta_j | \frac{\sum_i z_i x_{ij} - \frac{1}{2\lambda}}{\sum_i x_{ij}^2}, \sigma^2 (\sum_i x_{ij}^2)^{-1}\right).
\end{aligned}$$

Once we sample c_j as described above, we sample β_j from $p(\beta_j|\boldsymbol{\beta}_{-j}, \mathbf{c}, \mathbf{y}, \mathbf{X}, \sigma^2)$ in Equation (3). When $c_j = 0$, we set β_j to 0. If $c_j = 1$, we re-write the conditional probability distribution as

$$p(\beta_j|\boldsymbol{\beta}_{-j}, \mathbf{c}, \mathbf{y}, \mathbf{X}, \sigma^2) = \frac{p(\mathbf{y}|\boldsymbol{\beta}, \mathbf{c}, \mathbf{X}, \sigma^2) p(\beta_j)}{\int p(\mathbf{y}|\boldsymbol{\beta}, \mathbf{c}, \mathbf{X}, \sigma^2) p(\beta_j) \mathrm{d}\beta_j}. \quad (5)$$

We find that the denominator of Equation (5) is the same as what we computed in Equation (4). In fact, sampling from Equation (5) is equivalent to sampling from a mixture distribution of two components given as

$$m_j \sim \text{Bernoulli}\left(\frac{A_{(-)}}{A_{(-)}+A_{(+)}}\right)$$
$$\beta_j \sim \begin{cases} N_{(-),\beta_j<0} & \text{if } m_j = 0 \\ N_{(+),\beta_j>0} & \text{if } m_j = 1. \end{cases} \quad (6)$$

Using Equation (6), we augment β_j with m_j, and sample (β_j, m_j) by first drawing the mixture component label m_j from the Bernoulli distribution and then drawing β_j conditional on the m_j.

The conditional posterior for σ^2 is given as an inverse gamma distribution

$$\sigma^2|\boldsymbol{\beta}, \mathbf{c}, \mathbf{y}, \mathbf{X} \sim \text{Inv-gamma}\big((N+2J+\nu_0)/2, \\ \big(\sum_i (\mathbf{y}_i - \mathbf{X}_i\boldsymbol{\beta})^2 + \frac{1}{\lambda}\sum_j |\beta_j| + \nu_0 s_0^2\big)/2\big).$$

Next, we sample the parameters π_0 and π_1 of the transition matrix Π. The conditional posterior for π_0 is

$$\begin{aligned} p(\pi_0|\mathbf{c}) &\propto P(\mathbf{c}|\pi_0)p(\pi_0) \\ &= \prod_{k\in S_{00}} (e^{-d_k\rho_k} + (1-e^{-d_k\rho_k})\pi_0) \\ &\quad \cdot \prod_{k\in S_{01}} ((1-e^{-d_k\rho_k})(1-\pi_0)) \\ &\quad \cdot \pi_0^{a_{00}-1}(1-\pi_0)^{b_{00}-1} \\ &\propto \left(\pi_0^{n_{00}+a_{00}-1}(1-\pi_0)^{n_{01}+b_{00}-1}\right) \end{aligned} \quad (7)$$

where $S_{ml} = \{k|c_{k-1} = m, c_k = l\}$ and n_{ml} is the number of transitions from $c_{j-1} = m$ to $c_j = l$ in $\mathbf{c}$. We approximate n_{00} as

$$n_{00} = \sum_j \frac{(1-e^{-d_j\rho_j})\pi_0'}{e^{-d_j\rho_j} + (1-e^{-d_j\rho_j})\pi_0'} I(c_{j-1}=0, c_j=0),$$

where π_0' is the value from the previous sampling iteration, assuming that the number of events ($c_{j-1} = 0, c_j = 0$) due to $e^{-d_j\rho_j}$ and $(1-e^{-d_j\rho_j})\pi_0$ are proportional to their probabilities. In Equation (7), the conditional posterior for π_0 is Beta($n_{00}+a_{00}, n_{01}+b_{00}$) for π_0. Similarly, we obtain the conditional posterior for π_1 as Beta($n_{11}+a_{10}, n_{10}+b_{10}$).

Finally, we sample λ of the Laplacian prior from its conditional posterior

$$p(\lambda|\boldsymbol{\beta}, \sigma^2, \alpha, \gamma) = \text{Inv-gamma}\Big(J' + \alpha, \frac{\sum_{j\in S_{J'}} |\beta_j|}{2\sigma^2} + \gamma\Big),$$

where J' is the number of covariates with $c_j = 1$ in the current sampling iteration, and $S_{J'}$ is the set of such covariates.

3 EXPERIMENTS

We demonstrate our proposed model on simulated data and mouse data, and compare the performance with those from the model with independent Bernoulli prior for c_j's, ridge regression, and the lasso. We use ridge regression instead of ordinary least squares regression to prevent the singularity in matrix inversion, since often $J > N$ in our experiments. The regularization parameter in the ridge regression is set to a small value 0.1. The regularization parameter of the lasso is selected using a cross-validation.

In all of our experiments, for the block-regularized regression and the model with Bernoulli prior, we run the sampling algorithm for 5000 iterations after 2000 burn-in iterations. Samples are taken every 10 iterations. In the block-regularized regression, the priors for π_0 and π_1 are set to Beta(10,2), weakly encouraging the c_j's to stay in the same state as the c_{j-1}'s. Similarly, in the model with Bernoulli prior, the prior for the parameter p of the Benoulli distribution is set to Beta(10,2).

3.1 SIMULATION

We generate 360 haplotypes of a 40kb region with mutation rate 0.8/kb and recombination rate 0.1/kb using the software *ms* (Hudson 2002), and retain only those SNPs whose minor allele frequency is greater than 0.01. In the haplotypes generated under this setting, there are 108 SNPs in the region and 23 blocks in which no recombination events occurred. Then, we randomly pair two haplotypes to obtain genotypes of 180 individuals. We run the widely used software *Phase 2.1.1* (Li and Stephens 2003) on these data to estimate the recombination rate ρ between each pair of adjacent SNPs. We select 10 SNPs as relevant variables that are grouped into three blocks of 3, 2, and 5 SNPs respectively such that the SNPs within a group are located within a block of no recombination given by *ms*. Based on these selected causal SNPs, we generate a phenotype for each individual with $\beta_j = 2.5$ for causal SNPs and $\beta_j = 0$ for non-causal SNPs. The random noise generated from $N(0,1)$ is added to the simulated phenotype. The true parameters used in this simulation are shown in Figure 2(a). The dots show values for β_j's, and the line indicates the locations of the causal SNPs, where 1's represent causal SNPs, and 0's non-causal SNPs.

Using the data simulated from the true parameters in Figure 2(a), we fit our proposed model, the model with independent Bernoulli prior, ridge regression and the lasso, and plot the estimated $\boldsymbol{\beta}$ in Figures 2(b)-(e). For the block-regularized regression and the model with independent Bernoulli prior, we show the sam-

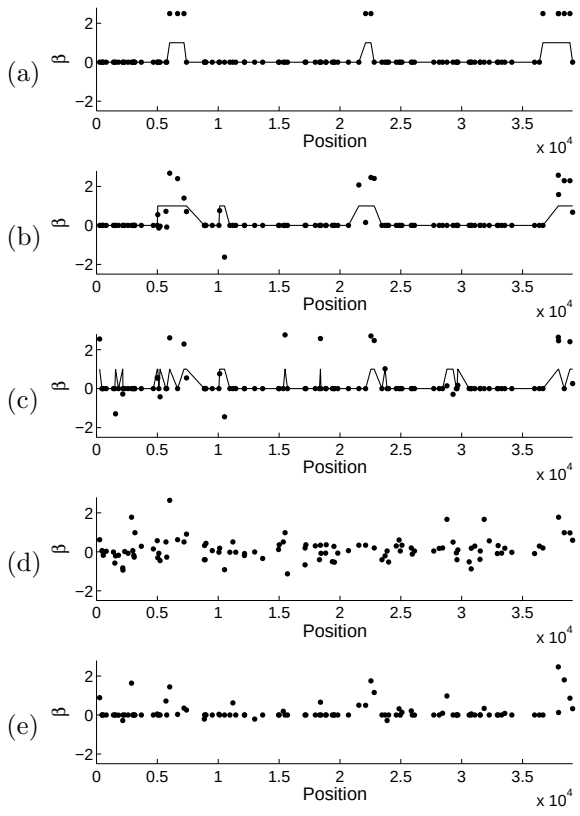


Figure 2: Simulation results with ρ=0.1/kb and $\beta_j = 2.5$ for relevant variables. True parameters are shown in (a), and the estimated parameters are shown for (b) the block-regularized regression, (c) the model with independent Bernoulli prior, (d) ridge regression, and (e) the lasso. Dots indicate β_j's at each position, and the lines in (a), (b), and (c) represent c_j's.

ple corresponding to the lowest train error, and plot the estimated $\mathbf{c}$ for the same sample as a line. The block-regularized regression in Figure 2(b) discovers the block structure in covariates with four groups of relevant variables, three of which roughly correspond to the three groups in the true parameters. Throughout our simulation experiments, we found that the groups of causal SNPs indicated in c_j's estimated by the block-regularized regression tend to extend to a slightly larger interval than in the true parameters, and that a subset of such SNPs with $c_j = 1$ has a large value for β_j. Thus, we can view c_j's as suggesting regions of relevant variables, and β_j's as deciding how relevant the variables with $c_j = 1$ are. As we see in Figures 2(c)-(e), the block structure is not obvious in the results from the other three methods.

Figures 3(a) and (b) show the estimated $P(c_j)$'s for the block-regularized regression and the model with independent Bernoulli prior respectively, using the same data as in Figure 2. Each $P(c_j)$ is estimated as the proportion of the number of times that the c_j is set to

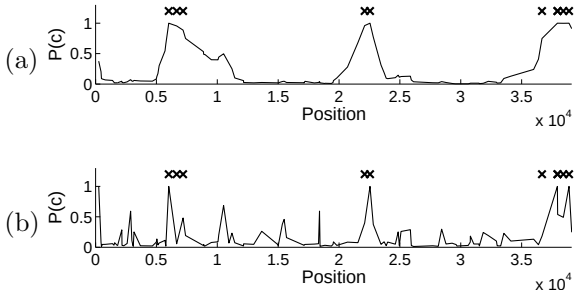


Figure 3: Estimated $P(c_j)$'s for (a) block-regularized regression and (b) the model with independent Bernoulli prior, corresponding to the results in Figure 2(b) and (c) respectively. The ×'s indicate the locations of true relevant variables.

Table 1: Summary Statistics of Simulated Data

ρ	Number of SNPs			Average number of SNPs per block
	Min	Max	Mean	
0.05/kb	86	260	150.8	5.59
0.1/kb	103	226	147.2	5.53
0.5/kb	99	214	151.0	1.18
1.0/kb	110	197	152.2	0.57

1 in samples for the c_j. The locations of the true causal SNPs are marked with ×'s. The block-regularized regression encourages a block structure among relevant and irrelevant covariates, leading to a smoother variation in $P(c_j)$'s between adjacent covariates compared to the model with independent Bernoulli prior.

In order to quantify the performance of the block-regularized regression and various other regression methods in the presence of different level of correlation among SNPs, we repeat the above procedure of simulating data for four different recombination rates ρ=0.05/kb, 0.1/kb, 0.5/kb, and 1/kb, using $\beta_j = 2.0$ for causal SNPs and $\beta_j = 0$ for non-causal SNPs. A lower recombination rate results in a more tightly linked SNPs and a stronger block structure in covariates. For each recombination rate, we generate 50 datasets of 180 individuals, and report the results averaged over these 50 datasets for the given recombination rate. Because of the random nature in the data generation process of the simulation software *ms*, the number of SNPs vary in each dataset even if the same parameters are used in simulation. The minimum, maximum, and average number of covariates in the 50 datasets for each recombination rate are shown in Table 1 as well as the average number of SNPs within a block given by *ms* during simulation. As the recombination rate gets higher, the correlations between adjacent SNPs become weaker, leading to only less than 1 SNP per block in the case of $\rho = 1.0$/kb.

Given these datasets generated as described above, we estimate $\boldsymbol{\beta}$ with the block-regularized regression, the

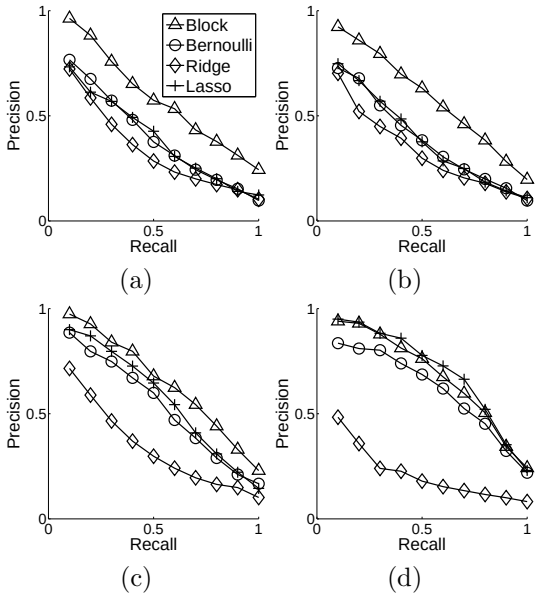


Figure 4: Precision-recall graphs for simulated data. (a) $\rho = 0.05$, (b) ρ=0.1, (c) ρ=0.5, and (d) ρ=1.0/kb.

model with independent Bernoulli prior, ridge regression, and the lasso, and plot precision-recall graphs in Figure 4(a)-(d) for recombination rates ρ=0.05/kb, 0.1/kb, 0.5/kb, and 1.0/kb respectively. To obtain each precision-recall curve in Figure 4, we estimate β_j's given a dataset and a regression method, and rank the SNPs according to the absolute values of β_j's. The SNP with the largest value of $|\beta_j|$ is considered as the most relevant. We compare the rankings of SNPs given by each regression method to the list of true causal SNPs, and compute the precisions and recalls shown in Figure 4. As can be seen in Figures 4(a) and (b), the block-regularized regression clearly outperforms other methods, since the correlations among SNPs are relatively high, and the block-regularized regression takes advantage of this correlation structure. As the recombination rate increases in Figures 4(c) and (d), the advantage of having a block model decreases. When ρ=1/kb, the average number of SNPs per block is less than 1 as shown in Table 1, and the block-regularized regression, the model with independent Bernoulli prior and the lasso perform similarly. We can use $P(c_j)$'s instead of $|\beta_j|$'s to rank the SNPs. When we plotted the precision-recall graphs according to the $P(c_j)$'s, we obtained similar results to the ones in Figure 4.

In Figure 5, we fit the block-regularized regression model to the dataset simulated with different values of β_j for relevant variables and the noise distributed as $N(0, 1)$, and plot the precision-recall graphs. The recombination rate ρ is set to 0.5/kb, and each precision-recall curve is the result averaged over 50 datasets. We see that as the signal to noise ratio increases, the performance increases.

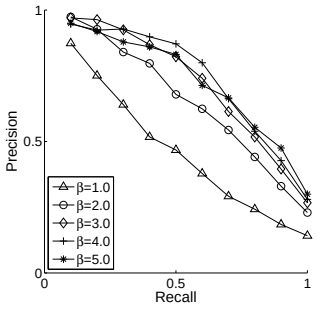


Figure 5: Precision-recall graphs for block-regularized regression, using simulated data with varying β_j's for relevant variables.

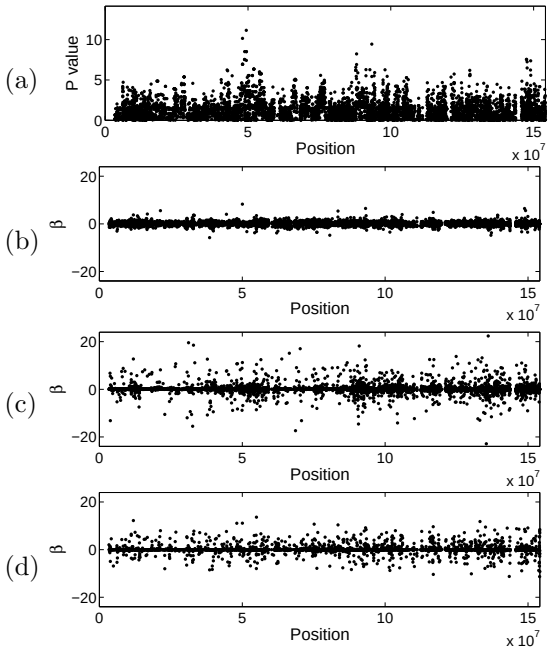


Figure 6: Results for the mouse haplotype data (chromosome 4) and the measurements of drinking preference. (a) $-\log(p$ value), (b) block-regularized regression, (c) the model with independent Bernoulli prior, (d) the lasso.

3.2 MOUSE DATA

We apply the block-regularized regression to the inbred laboratory mouse haplotype map publicly available from BROAD institue website[1]. We use the measurements for the sodium intake of 25% NaCl concentration for female mice (Tordoff et al. 2007) as phenotype. The data for 33 strains are available for both the genotype and phenotype dataset. Thus, the number of individuals in this experiment is 33. We consider the 8217 SNPs in chromosome 4 of length 154Mb as covariates. We are interested in scanning the chromosome and discovering the SNPs with high genetic effects on the phenotype.

[1]http://www.broad.mit.edu/mouse/hapmap

The most commonly used method for discovering SNPs highly associated with the phenotype is to perform a statistical test for the phenotype and one SNP at a time and report the SNPs with high p-values as significant. The result for the mouse data using one such test, the Wald test, is shown in Figure 6(a). The y-axis shows $-\log(p\text{-value})$ for each SNP. In Figure 6(b)-(d), we show the estimated $\boldsymbol{\beta}$ using the block-regularized regression, the model with independent Bernoulli prior, and the lasso respectively. For these three regression models, we divide the whole sequence into segments of 200 SNPs, and fit the model to one segment at a time. We see that the SNPs with high values of $-\log(p\text{-value})$ in Figure 6(a) roughly correspond to the SNPs with high β_j values in Figure 6(b).

4 CONCLUSIONS

In this paper, we considered the problem of finding a subset of covariates in a high-dimensional space that affect the output variable when there is a block structure in the covariates. In the context of association mapping, we proposed a regression-based model with a Markov chain prior that encodes the information in the correlation structure such as distance and recombination rate between adjacent SNP markers. We demonstrated on the simulated and mouse data that our proposed algorithm can be used to identify groups of SNP markers as a relevant block of causal SNPs.

The idea of representing the correlation structure as a Markov chain in a variable selection method to learn grouped relevant variables can be generalized to use a graphical model as a prior in a variable selection problem to represent an arbitrary correlation structure in variables in a high-dimensional space. Another interesting extension of the model is to model a structure in output variables as well when measurements of multiple output variables are available.

Acknowledgements

This research was supported by NSF Grants CCF-0523757 and DBI-0546594.

References

V. Cheung, R. Spielman, K. Ewens, T. Weber, M. Morley, and J. Burdick (2005). Mapping determinants of human gene expression by regional and genome-wide association. *Nature* **437**:1365-1369.

P. Fearnhead and P. Donnelly (2001). Estimating recombination rates from population genetic data. *Genetics* **159**:1299-1318.

E.I. George and R.E. McCulloch (1993). Variable selection via Gibbs sampling. *Journal of the American Statistical Association* **88**:881-889.

The International HapMap Consortium (2005). A haplotype map of the human genome. *Nature* **437**:1399-1320.

H. Ishwaran and J.S. Rao (2005). Spike and slab variable selection: frequentist and Bayesian strategies. *The Annals of Statistics* **33**(2):730-773.

R.R. Hudson (2002). Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics* **18**:337-338.

N. Li and M. Stephens (2003). Modelling linkage disequilibrium, and identifying recombination hotspots using snp data. *Genetics* **165**:2213-2233.

D.J. Nott and P.J. Green (2004). Bayesian variable selection and the Swendsen-Wang algorithm. *Journal of Computational and Graphical Statistics* **13**:141-157.

T. Park and G. Casella (2008). The Bayesian Lasso. (unpublished manuscript).

B. Servin, and M. Stephens (2007). Imputation-based analysis of association studies: candidate regions and quantitative traits. *PLoS Genetics* **3**(7):1296-1308.

K. Sohn and E.P. Xing (2007). Spectrum: joint Bayesian inference of population structure and recombination event. *The Fifteenth International Conference on Intelligence Systems for Molecular Biology.*

B. Stranger, M. Forrest, A. Clark, M. Minichiello, S. Deutsch, R. Lyle, S. Hunt, B. Kahl, S. Antonarakis, S. Tavare, P. Deloukas, and E. Dermitzakis (2005). Genome-wide associations of gene expression variation in humans. *PLoS Genetics* **1**(6):695-704.

R. Tibshirani (1996). Regression shrinkage and selection via the lasso. *Journal of Royal Statistical Society, Series B* **58**(1):267-288.

R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight (2005). Sparsity and smoothness via the fused lasso. *Journal of Royal Statistical Society, Series B* **67**(1):91-108.

M.G. Tordoff, A.A. Bachmanov, and D.R. Reed (2007). Forty mouse strain survey of water and sodium intake. *Physiology & Behavior* **91**(5):620-31.

M. Wainwright, P. Ravikumar, and J. Lafferty (2006). High-dimensional graphical model selection using L_1-regularized logistic regression. *Advances in Neural Information Processing Systems, 19.*

M. Yuan and Y. Lin (2005). Efficient empirical Bayes variable selection and estimation in linear models. *Journal of the American Statistical Association* **100**(472):1215-1225.

M. Yuan and Y. Lin (2006). Model selection and estimation in regression with grouped variables. *Journal of Royal Statistical Society, Series B* **68**(1):49-67.

N. Zaitlen, H. Kang, E. Eskin, and E. Halperin (2007). Leveraging the hapmap correlation structure in association studies. *The American Journal of Human Genetics* **80**(4):683-91.

K. Zhang, P. Calabrese, M. Nordborg, and F. Sun (2002). Haplotype block structure and its applications to association studies: power and study design. *The American Journal of Human Genetics* **71**:1386-1394.

The Evaluation of Causal Effects in Studies with an Unobserved Exposure/Outcome Variable: Bounds and Identification

Manabu Kuroki
Department of Systems Innovation
Graduate School of Engineering Science
Osaka University
mkuroki@sigmath.es.osaka-u.ac.jp

Zhihong Cai
Department of Biostatistics
School of Public Health
Kyoto University
cai@pbh.med.kyoto-u.ac.jp

Abstract

This paper deals with the problem of evaluating the causal effect using observational data in the presence of an unobserved exposure/outcome variable, when cause-effect relationships between variables can be described as a directed acyclic graph and the corresponding recursive factorization of a joint distribution. First, we propose identifiability criteria for causal effects when an unobserved exposure/outcome variable is considered to contain more than two categories. Next, when unmeasured variables exist between an unobserved outcome variable and its proxy variables, we provide the tightest bounds based on the potential outcome approach. The results of this paper are helpful to evaluate causal effects in the case where it is difficult or expensive to observe an exposure/outcome variable in many practical fields.

1 INTRODUCTION

The evaluation of causal effects from observational studies is one of the central aims in many fields of practical science. For this purpose, many researchers have attempted to clarify cause-effect relationships and to evaluate the causal effect of an exposure variable on an outcome variable through observed data. Statistical causal analysis, which is one of powerful tools for solving these problems, started with path analysis (Wright, 1923, 1934), and advanced to structural equation models (Wold, 1954; Bollen, 1989). It also has been modified in order to be applicable to categorical data (Goodman, 1973, 1974a, 1974b; Hagenaars, 1993). Recently, Pearl (2000) developed a new framework of causal modeling based on a directed acyclic graph and the corresponding nonparametric structural equation model.

In observational studies, there often exist unobserved variables, which makes it difficult to evaluate reliable causal effects. Many researchers have proposed various useful approaches to evaluate causal effects when unobserved variables are confounding factors between an exposure variable and an outcome variable, such as the instrumental variable method and sensitivity analysis. In the context of graphical causal models, Pearl (2000) provided the mathematical definition of the causal effect. In addition, when both an exposure variable and an outcome variable are observed, Pearl (2000), Tian and Pearl (2002) and Shpitser and Pearl (2006) discussed several graphical identification conditions for causal effects, which enable us to recognize situations where the causal effects can be evaluated from observational data.

However, in some situations, even an exposure/outcome variable is unobserved. For example, in a study to examine whether the socioeconomic gradient has an influence on low birth-weight, socioeconomic status is measured by some proxy variables such as income, wealth, education and occupation, since the true socioeconomic status is unobserved (Finch, 2003). Another example concerning an unobserved exposure is in occupational settings. Many epidemiological studies have addressed the question of carcinogenicity in workers exposed to diesel exhaust and coal mine dust, and most showed a low-to-medium increase in the risk of lung cancer. However, exposure measurement in these studies is mainly inferred on the basis of job classifications and may lead to misclassification (Hoffmann and Jockel, 2006). On the other hand, as an example concerning an unobserved outcome, Fleiss et al. (1976) reported a comparative clinical trial of ibuprofen, aspirin and placebo in the relief of post-extraction pain. Since the true outcome (pain relief) is unobserved, they used the Ridit analysis (Bross, 1958) to divide patients into five categories of pain relief: none, poor, fair, good and very good. These examples

show the importance of evaluating causal effects when an exposure/outcome variable is unobserved.

Kuroki et al. (2005) pointed out that it is difficult to apply the identification criteria proposed by Pearl and his colleagues to evaluate causal effects in such situations, and provided the graphical identifiability criteria when an unobserved exposure/outcome variable is continuous. In addition, Kuroki (2007) arranged the identification conditions proposed by Kuroki et al. (2005) to the case where an exposure/outcome variable is dichotomous. However, in many situations, researchers and practitioners are more interested in the different exposure levels (e.g., none, low, medium and high) than the pure binary exposure (exposed vs. unexposed), and are also more interested in the response levels (e.g., none, poor, fair, good and very good) than the simple binary response (improved vs. not improved).

Then, the main purpose of this paper is to provide identifiability criteria for causal effects from observational studies in the presence of an unobserved exposure/outcome variable with more than two categories. It will be shown that if we can observe some proxy variables that are affected by the unobserved variable, then the causal effect can be evaluated by using statistical causal analysis. More generally, we consider the case where there exist unmeasured variables between the unobserved exposure/outcome variable and its proxy variables. Under such a situation, the causal effect is not identifiable but the bounds on the causal effect can be derived. Finally, we illustrate our results with an example about social science.

2 PRELIMINARIES

2.1 BAYESIAN NETWORKS

Let $f(v_1, v_2, \ldots, v_n)$ be a strictly positive joint distribution of a set $\boldsymbol{V} = \{V_1, V_2, \cdots, V_n\}$ of variables, $f(v_i|v_j)$ the conditional distribution of V_i given $V_j = v_j$ $(V_i, V_j \in \boldsymbol{V})$ and $f(v_i)$ the marginal distribution of V_i. Similar notations are used for other distributions. For graph theoretic terminology used in this paper, refer to Kuroki et al. (2005).

Suppose that a set $\boldsymbol{V}$ of variables and a directed acyclic graph $G = (\boldsymbol{V}, \boldsymbol{E})$ are given. When the joint distribution of $\boldsymbol{V}$ is factorized recursively according to the graph G as the following equation, the graph is called a Bayesian network:

$$f(v_1, v_2, \cdots, v_n) = \prod_{i=1}^{n} f(v_i|\mathrm{pa}(v_i)). \qquad (1)$$

When $\mathrm{pa}(v_i)$ is an empty set, $f(v_i|\mathrm{pa}(v_i))$ is the marginal distribution $f(v_i)$ of v_i.

If a joint distribution is factorized recursively according to the graph G, the conditional independencies implied by the factorization (1) can be obtained from the graph G according to the d-separation criterion (Pearl, 1988), that is, if $\boldsymbol{Z}_1$ d-separates $\boldsymbol{Z}_2$ from $\boldsymbol{Z}_3$ in a directed acyclic graph G $(\boldsymbol{Z}_1, \boldsymbol{Z}_2, \boldsymbol{Z}_3 \subset \boldsymbol{V})$, then $\boldsymbol{Z}_2$ is conditionally independent of $\boldsymbol{Z}_3$ given $\boldsymbol{Z}_1$ in the corresponding recursive factorization (1); See, for example, Geiger et al. (1990).

2.2 CAUSAL EFFECT

Pearl (2000) defined a causal effect as a distribution of an outcome variable when conducting an external intervention, where an 'external intervention' means that a variable is forced to take on some fixed value, regardless of the values of other variables. If the distribution of the remaining variables represented in the directed acyclic graph remains essentially unchanged by such an external intervention, then the graph can be regarded as a causal diagram and the effect of the external intervention can be calculated from the joint factorized distribution. The exact definition is given as follows.

DEFINITION 1
Let $\boldsymbol{V} = \{X, Y\} \cup \boldsymbol{Q}$ $(\{X, Y\} \cap \boldsymbol{Q} = \phi)$ be a set of variables represented in a Bayesian network G. If the distribution of Y after setting X to a value x is given by

$$f(y|\mathrm{set}(X = x)) = \sum_{\boldsymbol{q}} \frac{f(x, y, \boldsymbol{q})}{f(x|\mathrm{pa}(x))}, \qquad (2)$$

then G is called a causal diagram with regard to X and equation (2) is called a causal effect of X on Y. Here, $\mathrm{set}(X = x)$ means that X is set to a value x by an external intervention. □

If Definition 1 holds true with regard to all pairs of variables in the graph, then the whole graph is said to be causal. For more details about the relationship between Bayesian networks and causal diagrams, see Pearl (2000).

Given a causal diagram G, in order to evaluate the causal effect $f(y|\mathrm{set}(X = x))$ of X on Y from a joint factorized distribution of observed variables, it is required to observe not only X and Y but also a set $\boldsymbol{Z}$ of other variables, such as confounders. Pearl (2000) provided 'the back door criterion' as one of graphical identifiability criteria for causal effects $f(y|\mathrm{set}(X = x))$, where 'identifiable' means that $f(y|\mathrm{set}(X = x))$ can be determined uniquely from a joint distribution of observed variables.

DEFINITION 2
Suppose that X is a non-descendant of Y in a directed acyclic graph G. If a set $\boldsymbol{Z}$ of vertices satisfies the

following conditions relative to an ordered pair (X, Y) of vertices, then $\boldsymbol{Z}$ is said to satisfy the back door criterion relative to (X, Y):

(i) no vertex in $\boldsymbol{Z}$ is a descendant of X;

(ii) $\boldsymbol{Z}$ blocks every path between X and Y that contains an arrow pointing to X. □

If a set $\boldsymbol{Z}$ of variables satisfies the back door criterion relative to (X, Y), then the causal effect $f(y|\text{set}(X = x))$ of X on Y is identifiable through the observation of $\boldsymbol{Z} \cup \{X, Y\}$ and is given by the formula

$$f(y|\text{set}(X = x)) = \sum_{\boldsymbol{z}} f(y|x, \boldsymbol{z}) f(\boldsymbol{z}). \tag{3}$$

When the back door criterion can not be applied to evaluate causal effects, Pearl (2000) provided 'the front door criterion', which is as follows:

DEFINITION 3
Suppose that X is a non-descendant of Y in a directed acyclic graph G. If a set $\boldsymbol{Z}$ of variables satisfies the following conditions relative to an ordered pair (X, Y) of variables, then $\boldsymbol{Z}$ is said to satisfy the front door criterion relative to (X, Y):

(i) $\boldsymbol{Z}$ blocks all directed paths from X to Y;

(ii) an empty set blocks every path between X and $\boldsymbol{Z}$ that contains an arrow pointing to X;

(iii) X blocks every path between any vertex in $\boldsymbol{Z}$ and Y. □

If a set $\boldsymbol{Z}$ of variables satisfies the front door criterion relative to (X, Y), then the causal effect $f(y|\text{set}(X = x))$ of X on Y is identifiable through the observation of $\boldsymbol{Z} \cup \{X, Y\}$ and is given by the formula

$$f(y|\text{set}(X = x)) = \sum_{x', \boldsymbol{z}} f(y|x', \boldsymbol{z}) f(\boldsymbol{z}|x) f(x'). \tag{4}$$

3 IDENTIFICATION OF CAUSAL EFFECTS

In section 2, it is assumed that both an exposure variable and an outcome variable are observable. If either of them is unobserved, we cannot identify the causal effect of an exposure on an outcome even if a set of variables satisfying the back door criterion or the front door criterion are observed. In this section, we consider the case where an unobserved exposure/outcome variable is assumed to be discrete. Let X be an exposure variable and Y be an outcome variable. Though X or Y is unobserved, researchers are interested in dividing them into k categories. For example, when the domain of Y is divided into $k = 3$ categories, y_1, y_2 and y_3 may represent the poor, fair and good response levels. Then, let U be either X or Y which is an unobserved variable ($u \in \{u_1, \cdots, u_k\}$). In addition, let a set $\boldsymbol{S}$ and a set $\boldsymbol{T}$ be observed proxy variables that are affected by the unobserved variable U. Assume that we can select k distinct vectors from the domains of a set $\boldsymbol{S}$ and a set $\boldsymbol{T}$ of variables, denoted as $\boldsymbol{t}_1, \cdots, \boldsymbol{t}_k$ and $\boldsymbol{s}_1, \cdots, \boldsymbol{s}_k$, respectively. A set $\boldsymbol{W}$ and a set $\boldsymbol{Z}$ are assumed to be continuous and/or discrete variables. Furthermore, let P and Q be k dimensional nonsingular matrices such that

$$P = \begin{pmatrix} 1 & f(\boldsymbol{t}_1|\boldsymbol{z}) & \cdots & f(\boldsymbol{t}_{k-1}|\boldsymbol{z}) \\ f(\boldsymbol{s}_1|\boldsymbol{z}) & f(\boldsymbol{s}_1, \boldsymbol{t}_1|\boldsymbol{z}) & \cdots & f(\boldsymbol{s}_1, \boldsymbol{t}_{k-1}|\boldsymbol{z}) \\ \vdots & \vdots & \vdots & \vdots \\ f(\boldsymbol{s}_{k-1}|\boldsymbol{z}) & f(\boldsymbol{s}_{k-1}, \boldsymbol{t}_1|\boldsymbol{z}) & \cdots & f(\boldsymbol{s}_{k-1}, \boldsymbol{t}_{k-1}|\boldsymbol{z}) \end{pmatrix}, \tag{5}$$

$$Q = \begin{pmatrix} f(\boldsymbol{w}|\boldsymbol{z}) & f(\boldsymbol{w}, \boldsymbol{t}_1|\boldsymbol{z}) & \cdots & f(\boldsymbol{w}, \boldsymbol{t}_{k-1}|\boldsymbol{z}) \\ f(\boldsymbol{w}, \boldsymbol{s}_1|\boldsymbol{z}) & f(\boldsymbol{w}, \boldsymbol{s}_1, \boldsymbol{t}_1|\boldsymbol{z}) & \cdots & f(\boldsymbol{w}, \boldsymbol{s}_1, \boldsymbol{t}_{k-1}|\boldsymbol{z}) \\ \vdots & \vdots & \vdots & \vdots \\ f(\boldsymbol{w}, \boldsymbol{s}_{k-1}|\boldsymbol{z}) & f(\boldsymbol{w}, \boldsymbol{s}_{k-1}, \boldsymbol{t}_1|\boldsymbol{z}) & \cdots & f(\boldsymbol{w}, \boldsymbol{s}_{k-1}, \boldsymbol{t}_{k-1}|\boldsymbol{z}) \end{pmatrix}. \tag{6}$$

Then, the following theorem is obtained.

THEOREM 1
Given a causal diagram G on $\boldsymbol{V}$ with $\boldsymbol{S} \cup \boldsymbol{T} \cup \{U\} \cup \boldsymbol{Z} \cup \boldsymbol{W} (\subset \boldsymbol{V})$, suppose that

(i) $\boldsymbol{Z} \cup \{U\}$ d-separates $\boldsymbol{S}$ from $\boldsymbol{T}$ and $\boldsymbol{W}$ from $\boldsymbol{S} \cup \boldsymbol{T}$;

(ii) $f(u_1|\boldsymbol{z}) < \cdots < f(u_k|\boldsymbol{z})$ holds true for any $\boldsymbol{z}$;

(iii) For the matrices defined as equations (5) and (6), both P and Q are k dimensional nonsingular matrices and $|Q - \lambda P| = 0$ has non-zero distinct solutions of λ $(0 < \lambda_1 < ... < \lambda_k)$ for any z $(P \neq Q)$,

then the distribution $f(u, \boldsymbol{w}, \boldsymbol{z})$ is identifiable through the observation of $\boldsymbol{S} \cup \boldsymbol{T} \cup \boldsymbol{Z} \cup \boldsymbol{W}$. □

PROOF OF THEOREM 1
Let

$$P_1 = \begin{pmatrix} 1 & f(\boldsymbol{t}_1|u_1, \boldsymbol{z}) & \cdots & f(\boldsymbol{t}_{k-1}|u_1, \boldsymbol{z}) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & f(\boldsymbol{t}_1|u_k, \boldsymbol{z}) & \cdots & f(\boldsymbol{t}_{k-1}|u_k, \boldsymbol{z}) \end{pmatrix},$$

$$P_2 = \begin{pmatrix} 1 & f(\boldsymbol{s}_1|u_1, \boldsymbol{z}) & \cdots & f(\boldsymbol{s}_{k-1}|u_1, \boldsymbol{z}) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & f(\boldsymbol{s}_1|u_k, \boldsymbol{z}) & \cdots & f(\boldsymbol{s}_{k-1}|u_k, \boldsymbol{z}) \end{pmatrix},$$

and $\Delta = \text{diag}(f(\boldsymbol{w}|u_1, \boldsymbol{z}), \cdots, f(\boldsymbol{w}|u_k, \boldsymbol{z}))$ be the k dimensional diagonal matrices of conditional probabili-

ties of observed variables $\boldsymbol{W}$ given U and $\boldsymbol{Z}$. In addition, let $M = \mathrm{diag}(f(u_1|\boldsymbol{z}), \cdots, f(u_k|\boldsymbol{z}))$ be the k dimensional diagonal matrix of conditional probabilities of U given $\boldsymbol{Z}$. Then, the followings are derived:

$$\begin{aligned} f(\boldsymbol{w}|\boldsymbol{z}) &= \sum_{i=1}^{k} f(\boldsymbol{w}|u_i, \boldsymbol{z}) f(u_i|\boldsymbol{z}), \\ f(\boldsymbol{t}_j, \boldsymbol{s}_l|\boldsymbol{z}) &= \sum_{i=1}^{k} f(\boldsymbol{s}_l|u_i, \boldsymbol{z}) f(\boldsymbol{t}_j|u_i, \boldsymbol{z}) f(u_i|\boldsymbol{z}), \end{aligned}$$

$$\begin{aligned} f(\boldsymbol{w}, \boldsymbol{t}_j, \boldsymbol{s}_l|\boldsymbol{z}) = \sum_{i=1}^{k} & f(\boldsymbol{w}|u_i, \boldsymbol{z}) f(\boldsymbol{t}_j|u_i, \boldsymbol{z}) \\ & \times f(\boldsymbol{s}_l|u_i, \boldsymbol{z}) f(u_i|\boldsymbol{z}) \end{aligned} \tag{7}$$

for $j, l = 1, 2, \cdots, k$. Then, we can obtain

$$P = P_2' M P_1 \quad \text{and} \quad Q = P_2' M \Delta P_1.$$

Thus, by noting that both P and Q are nonsingular matrices of conditional probabilities of observed variables, consider the following equation for λ:

$$\begin{aligned} |Q - \lambda P| &= |P_1' M \Delta P_2 - \lambda P_1' M P_2| \\ &= |P_1'||M||\Delta - \lambda I_k||P_2| = 0, \end{aligned} \tag{8}$$

where I_k is a k dimensional identity matrix. By solving equation (8), we can obtain the element $f(\boldsymbol{w}|u_i, \boldsymbol{z})$ $(i = 1, 2, \cdots, k)$ of Δ. Here, let λ_i be a disjoint solution of the above equation satisfying $0 < \lambda_1 < \cdots < \lambda_k$. This means that Δ is identifiable if the order of $f(\boldsymbol{w}|u_1, \boldsymbol{z}), \cdots, f(\boldsymbol{w}|u_k, \boldsymbol{z})$ is known. Let $E_i = \mathrm{diag}(\alpha_{1(i)}, \cdots, \alpha_{k(i)})$ be a k dimensional diagonal matrix $(i = 1, 2)$ and $A_1 = P_1^{-1} E_1$ be a k dimensional matrix. Since Δ, E_1 and M are diagonal matrices and the elements of Δ are correspondent to the solutions of equation (8), we can obtain

$$Q A_1 = P A_1 \Delta.$$

This means that matrix $A_1 = P_1^{-1} E_1$ is the solution of the characteristic equation

$$(Q - \lambda P)\boldsymbol{x} = \boldsymbol{0}_k$$

for $\boldsymbol{x}$, which indicates that A_1 is estimable ($\lambda \in \{\lambda_1, \cdots, \lambda_k\}$). Here, $\boldsymbol{0}_k$ is a k dimensional zero vector.

Since $P_1 = E_1 A_1^{-1}$, letting $A_1^{-1} = (a_1^{i,j})$, we can obtain

$$\begin{aligned} P_1 &= \begin{pmatrix} 1 & f(\boldsymbol{t}_1|u_1, \boldsymbol{z}) & \cdots & f(\boldsymbol{t}_{k-1}|u_1, \boldsymbol{z}) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & f(\boldsymbol{t}_1|u_k, \boldsymbol{z}) & \cdots & f(\boldsymbol{t}_{k-1}|u_k, \boldsymbol{z}) \end{pmatrix} \\ &= E_1 A_1^{-1} \\ &= \begin{pmatrix} \alpha_{1(1)} a_1^{1,1} & \alpha_{1(1)} a_1^{1,2} & \cdots & \alpha_{1(1)} a_1^{1,k} \\ \alpha_{2(1)} a_1^{2,1} & \alpha_{2(1)} a_1^{2,2} & \cdots & \alpha_{2(1)} a_1^{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{k(1)} a_1^{k,1} & \alpha_{k(1)} a_1^{k,2} & \cdots & \alpha_{k(1)} a_1^{k,k} \end{pmatrix}. \end{aligned}$$

Then, $\alpha_{i(1)} = 1/a_1^{i,1}$ can be uniquely obtained $(i = 1, 2, \cdots, k)$ from the first column, which indicates that P_1 is also estimable from $E_1 A^{-1}$ according to the order of $\lambda_1, \cdots, \lambda_k$, where $E_1 = \mathrm{diag}(1/a_1^{1,1}, \cdots, 1/a_1^{k,1})$.

On the other hand, letting $A_2 = P_2^{-1} E_2$, since Δ, E_2 and M are diagonal matrices and the elements of Δ are correspondent to the solutions of (8), we can obtain

$$Q' A_2 = P' A_2 \Delta.$$

This means that matrix $A_2 = P_2^{-1} E_2$ is the solution of the characteristic equation

$$(Q' - \lambda P')\boldsymbol{x} = \boldsymbol{0}_k$$

for $\boldsymbol{x}$. Thus, A_2 is also estimable ($\lambda \in \{\lambda_1, \cdots, \lambda_k\}$).

Since $P_2 = E_2 A_2^{-1}$, letting $A_2^{-1} = (a_2^{i,j})$, we can obtain

$$\begin{aligned} P_2 &= \begin{pmatrix} 1 & f(\boldsymbol{s}_1|u_1, \boldsymbol{z}) & \cdots & f(\boldsymbol{s}_{k-1}|u_1, \boldsymbol{z}) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & f(\boldsymbol{s}_1|u_k, \boldsymbol{z}) & \cdots & f(\boldsymbol{s}_{k-1}|u_k, \boldsymbol{z}) \end{pmatrix} \\ &= E_2 A_2^{-1} \\ &= \begin{pmatrix} \alpha_{1(2)} a_2^{1,1} & \alpha_{1(2)} a_2^{1,2} & \cdots & \alpha_{1(2)} a_2^{1,k} \\ \alpha_{2(2)} a_2^{2,1} & \alpha_{2(2)} a_2^{2,2} & \cdots & \alpha_{2(2)} a_2^{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{k(2)} a_2^{k,1} & \alpha_{k(2)} a_2^{k,2} & \cdots & \alpha_{k(2)} a_2^{k,k} \end{pmatrix}. \end{aligned}$$

Then, $\alpha_{i(2)} = 1/a_2^{i,1}$ can be uniquely obtained $(i = 1, 2, \cdots, k)$ from the first column, which indicates that P_2 is also estimable from $E_2 A_2^{-1}$ according to the order of $\lambda_1, \cdots, \lambda_k$, where $E_2 = \mathrm{diag}(1/a_2^{1,1}, \cdots, 1/a_2^{k,1})$. From these results, we can obtain

$$P_2'^{-1} P P_1^{-1} = P_2'^{-1} (P_2' M P_1) P_1^{-1} = M. \tag{9}$$

Thus, we can obtain the element $f(u_i|\boldsymbol{z})$ $(i = 1, 2, \cdots, k)$ of M from equation (9), which is determined uniquely according to the order of disjoint solution $\lambda_1 < \cdots < \lambda_k$ of equation (8). Inversely, since the order of the elements of M is identifiable from condition (ii), the order of $\lambda_1, \cdots, \lambda_k$ is identifiable. Thus, the conditional distribution of U given $\boldsymbol{z}$ is estimable through the observation of $\boldsymbol{S} \cup \boldsymbol{T} \cup \boldsymbol{W} \cup \boldsymbol{Z}$. Then, since

$$f(u, \boldsymbol{z}, \boldsymbol{w}) = f(\boldsymbol{w}|u, \boldsymbol{z}) f(u|\boldsymbol{z}) f(\boldsymbol{z}),$$

$f(u, \boldsymbol{z}, \boldsymbol{w})$ is estimable through the observation of $\boldsymbol{S} \cup \boldsymbol{T} \cup \boldsymbol{W} \cup \boldsymbol{Z}$. Q.E.D.

Based on Theorem 1, the following corollary can be derived immediately.

COROLLARY 1

Suppose that one element of $\{X, Y\}$ is an unobserved variable U and the other element is included in a

set $\boldsymbol{Z}\cup\boldsymbol{W}$ of observed variables. Let $\boldsymbol{C}$ be a subset of $\boldsymbol{Z}\cup\boldsymbol{W}\backslash\{X,Y\}$ that satisfies the identifiability criteria for the causal effect $f(y|\mathrm{set}(X=x))$. If a set $\boldsymbol{Z}\cup\boldsymbol{W}\cup\boldsymbol{S}\cup\boldsymbol{T}$ of observed variables satisfies conditions (i)-(iii) in Theorem 1, then the causal effect $f(y|\mathrm{set}(X=x))$ is identifiable. □

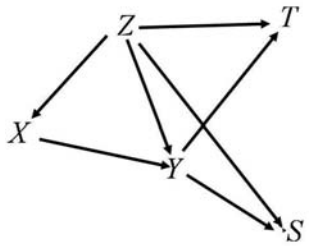


Fig. 1 : Causal diagram (1)

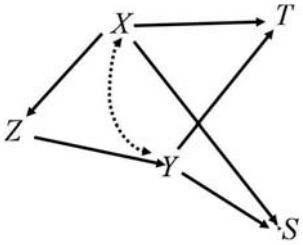


Fig. 2 : Causal diagram (2)

We use two examples to illustrate Corollary 1. First, consider the causal diagram shown in Fig. 1. Setting $\boldsymbol{W}$ in Corollary 1 to X in Fig. 1, we can recognize that $\{Z,Y\}$ d-separates S from T and X from $\{S,T\}$. In addition, $\boldsymbol{C}=\{Z\}$ satisfies the back door criterion relative to (X,Y). Then, based on the proof of Theorem 1, the distribution of X,Y and Z can be constructed according to the distribution of X,Z,S and T. Thus, if conditions (ii) and (iii) hold true, then the causal effect $f(y|\mathrm{set}(X=x))$ is identifiable through the observation of X,S,T and Z. The closed form expression in the case where Y is a dichotomous variable is provided in Kuroki (2007).

Next, consider the causal diagram shown in Fig. 2, where the back door criterion cannot be applied to identify the causal effect of $f(y|\mathrm{set}(X=x))$, because there is a bi-directed arrow in Fig. 2 which indicates that there exist some unmeasured confounders between X and Y. Letting U, $\boldsymbol{W}$ and $\boldsymbol{Z}$ in Corollary 1 be Y, ϕ and X in Fig. 2, we can recognize that $\{X,Y\}$ d-separates S from T and Z from $\{S,T\}$. In addition, $\boldsymbol{C}=\{Z\}$ satisfies the front door criterion relative to (X,Y). Then, based on the proof of Theorem 1, the distribution of X,Y and Z can be constructed according to the distribution of X,Z,S and T. Thus, if conditions (ii) and (iii) hold true, then the causal effect $f(y|\mathrm{set}(X=x))$ is identifiable through the observation of X,S,T and Z. This example shows that our result can also be applied to situations where there is no variable that satisfies the back door criterion.

When identifying the causal effect using Theorem 1, it is required that $f(u_1|\boldsymbol{z})<f(u_2|\boldsymbol{z})<\cdots<f(u_k|\boldsymbol{z})$ holds true for any $\boldsymbol{z}$. If such an order information is not available, it is impossible to judge whether the causal effect is identifiable or not from Theorem 1. But we can evaluate the bounds of the causal effect. Consider the causal diagram shown in Fig. 1 as an example. By noting that $f(y|x,z)=f(x|y,z)f(y|z)/f(x|z)$ holds true, letting the diagonal elements of M be $m_1,\cdots,m_k$ determined according to the order $\lambda_1<\cdots<\lambda_k$, the bounds of the causal effect $f(y|\mathrm{set}(X=x))$ are

$$\sum_z \frac{\min_i\{\lambda_i m_i\}}{f(x|z)}f(z)\leq f(y|\mathrm{set}(X=x))$$

$$f(y|\mathrm{set}(X=x))\leq \sum_z \frac{\max_i\{\lambda_i m_i\}}{f(x|z)}f(z).$$

4 BOUNDS ON CAUSAL EFFECT

4.1 POTENTIAL OUTCOME APPROACH

In this section, we consider the case where there exist unmeasured variables between an unobserved outcome variable and its proxy variables. Under such a situation, it is impossible to identify the causal effect, but we can derive the bounds on the causal effect by using the potential outcome approach. For simplicity, we only consider the case of an unobserved dichotomous outcome variable, though our result can apply to multi-categorical case directly.

Let X and Y be a dichotomous exposure variable ($x\in\{x_0,x_1\}$) and a dichotomous outcome variable ($y\in\{y_0,y_1\}$). Then, the ith of the N subjects has both an outcome $Y_{x_1}(i)$ that have resulted if he was exposed to x_1, and an outcome $Y_{x_0}(i)$ that have resulted if he was exposed to x_0. When the N subjects in the study are considered as a random sample from the target population, since $Y_{x_1}(i)$ and $Y_{x_0}(i)$ can be referred to as the values of random variables Y_{x_1} and Y_{x_0} respectively, the causal effect can be defined as the probability $P(Y_x=y)\stackrel{\triangle}{=}f(y_x)$ of the potential outcome ($x\in\{x_0,x_1\}$). The potential outcome Y_x is observed only if the subject receives exposure x ($x\in\{x_1,x_0\}$). Thus, when randomized experiment is conducted and compliance is perfect, the causal effect of X on Y is

$$f(y_x)\stackrel{\triangle}{=}f(y|\mathrm{set}(X=x))=f(y|x), \qquad (10)$$

by using the consistency condition (Pearl, 2000)

$$X=x \;\Rightarrow\; Y_x=Y.$$

On the other hand, when randomized experiment is difficult to conduct and only observational data is available, we can still estimate the causal effect according to the strongly-ignorable-treatment-assignment (SITA) condition (Rosenbaum and Rubin, 1983). That is, for the exposure variable X, if there exists such a set $\boldsymbol{Z}$ of covariates that X is conditionally independent of (Y_{x_1},Y_{x_0}) given $\boldsymbol{Z}$, denoted as $X\perp\!\!\!\perp(Y_{x_1},Y_{x_0})\,|\,\boldsymbol{Z}$, we shall say treatment assignment is strongly ignorable given $\boldsymbol{Z}$, or $\boldsymbol{Z}$ satisfies the SITA condition. Thus, $f(y|\mathrm{set}(X=x))$ is estimable by using Z and is given as equation (3).

4.2 FORMULATION

In order to describe our problem, we consider the simple causal diagram shown in Fig. 3, where X, S and T are observed dichotomous variables ($x\in\{x_0, x_1\}, s\in\{s_0, s_1\}, t\in\{t_0, t_1\}$). In addition, Y is an unobserved dichotomous variable ($y\in\{y_0, y_1\}$). Furthermore, there is no confounder between X and Y, but there exist unmeasured variables between Y, S and T.

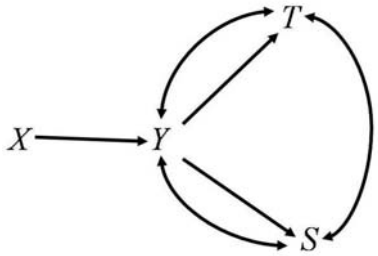


Fig. 3: Causal Diagram (3)

Then, the potential outcomes corresponding to this figure can be introduced as follows:

(i) Potential outcome R_t in the context of Y as an exposure and T as an outcome:

r_{t_0}:$(T_{y_0}, T_{y_1}) = (t_0, t_0)$,$r_{t_1}$:$(T_{y_0}, T_{y_1}) = (t_0, t_1)$,

r_{t_2}:$(T_{y_0}, T_{y_1}) = (t_1, t_0)$,$r_{t_3}$:$(T_{y_0}, T_{y_1}) = (t_1, t_1)$,

(ii) Potential outcome R_s in the context of Y as an exposure and S as an outcome:

$$r_{s_0}:(S_{y_0}, S_{y_1}) = (s_0, s_0), r_{s_1}:(S_{y_0}, S_{y_1}) = (s_0, s_1),$$
$$r_{s_2}:(S_{y_0}, S_{y_1}) = (s_1, s_0), r_{s_3}:(S_{y_0}, S_{y_1}) = (s_1, s_1),$$

(iii) Potential outcome R_y in the context of X as an exposure and Y as an outcome:

$$r_{y_0}:(Y_{x_0}, Y_{x_1}) = (y_0, y_0), r_{y_1}:(Y_{x_0}, Y_{x_1}) = (y_0, y_1),$$
$$r_{y_2}:(Y_{x_0}, Y_{x_1}) = (y_1, y_0), r_{y_3}:(Y_{x_0}, Y_{x_1}) = (y_1, y_1).$$

Letting $q_{ijk} = P(r_{t_i}, r_{s_j}, r_{y_k})$ be counterfactual probabilities $(i, j, k = 0, 1, 2, 3)$, these parameters are constrained by the probabilistic equality

$$\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} q_{ijk} = 1 \text{ and } 0\leq q_{ijk}\leq 1. \qquad (11)$$

Let $p_{ij\cdot k}$ be the observed conditional probabilities of $(T, S) = (t_i, s_j)$ given $X = x_k$, that is, $p_{ij\cdot k} = P(t_i, \boldsymbol{s}_j|x_k)$. These observed conditional probabilities impose the constraints by applying both the consistency condition and $X \perp\!\!\!\perp (S_{y_0}, S_{y_1}, T_{y_0}, T_{y_1}, Y_{x_0}, Y_{x_1})$ to the counterfactual probabilities:

$$p_{00\cdot 1} = \sum_{i=0,1}\sum_{j=0,1}\sum_{k=0,2} q_{ijk} + \sum_{i=0,2}\sum_{j=0,2}\sum_{k=1,3} q_{ijk},$$
$$p_{01\cdot 1} = \sum_{i=0,1}\sum_{j=2,3}\sum_{k=0,2} q_{ijk} + \sum_{i=0,2}\sum_{j=1,3}\sum_{k=1,3} q_{ijk}$$
$$p_{10\cdot 1} = \sum_{i=2,3}\sum_{j=0,1}\sum_{k=0,2} q_{ijk} + \sum_{i=1,3}\sum_{j=0,2}\sum_{k=1,3} q_{ijk},$$
$$p_{11\cdot 1} = \sum_{i=2,3}\sum_{j=2,3}\sum_{k=0,2} q_{ijk} + \sum_{i=1,3}\sum_{j=1,3}\sum_{k=1,3} q_{ijk}$$
$$p_{00\cdot 0} = \sum_{i=0,1}\sum_{j=0,1}\sum_{k=0,1} q_{ijk} + \sum_{i=0,2}\sum_{j=0,2}\sum_{k=2,3} q_{ijk},$$
$$p_{01\cdot 0} = \sum_{i=0,1}\sum_{j=2,3}\sum_{k=0,1} q_{ijk} + \sum_{i=0,2}\sum_{j=1,3}\sum_{k=2,3} q_{ijk}$$
$$p_{10\cdot 0} = \sum_{i=2,3}\sum_{j=0,1}\sum_{k=0,1} q_{ijk} + \sum_{i=1,3}\sum_{j=0,2}\sum_{k=2,3} q_{ijk},$$
$$p_{11\cdot 0} = \sum_{i=2,3}\sum_{j=2,3}\sum_{k=0,1} q_{ijk} + \sum_{i=1,3}\sum_{j=1,3}\sum_{k=2,3} q_{ijk}. \qquad (12)$$

Then, the quantities we wish to bound are:

$$f(y_1|\text{set}(X = x_1)) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=1,3} q_{ijk}, \qquad (13)$$

$$f(y_1|\text{set}(X = x_0)) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=2,3} q_{ijk}. \qquad (14)$$

Optimizing the functions (13) and (14), subject to equality constraints (11) and (12), defines a linear programming (LP) problem that lends itself to closed-form solution. Balke (1995) describes a computer program that takes symbolic description of LP problems and returns symbolic expressions for the desired bounds. The problem works by systematically enumerating the vertices of the constraint polygon of the dual problem. The bounds reported in this paper were produced by using Balke's program, and will be stated here without proofs; their correctness can be verified by manually enumerating the vertices as described in Balke (1995). These bounds are guaranteed to be sharp because the optimization is global.

Given the observed conditional probabilities, the constraints (11) and (12) induce the bounds $[0, 1]$, which indicates that the causal knowledge available from Fig. 3 can not provide useful evaluation of the causal effect. However, if we assume the monotonic assumption, which leads to

$$q_{2jk} = q_{i2k} = q_{ij2} = 0 \qquad i, j, k = 0, 1, 2, 3.$$

Then, we can obtain the tightest bounds on the causal effects:

$$0\leq f(y_1|\text{set}(X = x_0))\leq \min\left\{\begin{array}{l} p_{01\cdot 0} + p_{10\cdot 0} + p_{11\cdot 0} + p_{00\cdot 1} \\ p_{01\cdot 0} + p_{11\cdot 0} + p_{10\cdot 1} + p_{00\cdot 1} \\ p_{10\cdot 0} + p_{11\cdot 0} + p_{01\cdot 1} + p_{00\cdot 1} \\ p_{11\cdot 0} + p_{00\cdot 1} + p_{10\cdot 1} + p_{01\cdot 1} \end{array}\right\}, \qquad (15)$$

$$\max\left\{\begin{array}{c} p_{00\cdot 0}-p_{00\cdot 1} \\ p_{11\cdot 1}-p_{11\cdot 0} \\ p_{00\cdot 0}+p_{10\cdot 0}-p_{00\cdot 1}-p_{10\cdot 1} \\ p_{00\cdot 0}+p_{01\cdot 0}-p_{00\cdot 1}-p_{01\cdot 1} \end{array}\right\}$$
$$\leq f(y_1|\mathrm{set}(X=x_1))\leq 1. \quad (16)$$

It is noted that these formulas require two proxy variables S and T. If only one proxy variable is available, the tightest bounds on the causal effects become $[0,1]$, which shows that one proxy variable provides no useful information for evaluating the causal effect.

Finally, we consider a more complicated situation that there are confounders between X and Y. In the case, if we can observe a set $\boldsymbol{Z}$ of covariates that satisfy the SITA condition, and $X \perp\!\!\!\perp (S_{y_0}, S_{y_1}, T_{y_0}, T_{y_1}, Y_{x_0}, Y_{x_1})|\boldsymbol{Z}$ holds true, by the same procedure as the above, the bounds on the causal effects can be evaluated as

$$0\leq f(y_1|\mathrm{set}(X=x_0))\leq$$
$$\sum_z \min\left\{\begin{array}{c} p_{01\cdot 0z}+p_{10\cdot 0z}+p_{11\cdot 0z}+p_{00\cdot 1z} \\ p_{01\cdot 0z}+p_{11\cdot 0z}+p_{10\cdot 1z}+p_{00\cdot 1z} \\ p_{10\cdot 0z}+p_{11\cdot 0z}+p_{01\cdot 1z}+p_{00\cdot 1z} \\ p_{11\cdot 0z}+p_{00\cdot 1z}+p_{10\cdot 1z}+p_{01\cdot 1z} \end{array}\right\} P(z),$$

$$\sum_z \max\left\{\begin{array}{c} p_{00\cdot 0z}-p_{00\cdot 1z} \\ p_{11\cdot 1z}-p_{11\cdot 0z} \\ p_{00\cdot 0z}+p_{10\cdot 0z}-p_{00\cdot 1z}-p_{10\cdot 1z} \\ p_{00\cdot 0z}+p_{01\cdot 0z}-p_{00\cdot 1z}-p_{01\cdot 1z} \end{array}\right\} P(z)$$
$$\leq f(y_1|\mathrm{set}(X=x_1))\leq 1.$$

With the similar procedure above, we can also derive the bounds on the causal effect when there exist unmeasured variables between an unobserved exposure variable and its proxy variables.

5 EXAMPLE

We illustrate our results using the data from a political action study reported by Hagenaars (1993). He analyzed the data in order to evaluate the causal effect of education on political involvement. The variables of interest are the following:

X: education (1: some college; 2: less than college),

S: ideological level (1: ideologues; 2: nonideologues),

T: repression potential (1: low; 2: high),

Y: political involvement (1: high; 2: low),

Here, we concentrate our discussion on evaluating the causal effect of X on Y, where Y is unobserved. In order to help readers understand our results, we consider a submodel in Hagenaars (1993), which is shown in Fig. 3.

First, we consider the situation where there is no bidirected arrow in Fig. 3. Since Y d-separates any two vertices in $\{S,T,X\}$, and there is no unmeasured variables between X and Y, Theorem 1 can be used to achieve our aim. Because the real data of this model is not available, we generate hypothetical data according to Fig. 3, which is shown in Table 1.

Table 1. Hypothetical Data of the Example

		t_1	t_0
x_1	s_1	0.0648	0.0432
	s_0	0.1392	0.0528
x_0	s_1	0.1092	0.2478
	s_0	0.1568	0.1862

In this example, we suppose that $f(u_1) < f(u_2)$. Then, letting

$$P=\begin{pmatrix} 1.000 & f(t_1) \\ f(s_1) & f(s_1,t_1)\end{pmatrix}=\begin{pmatrix} 1.000 & 0.47 \\ 0.465 & 0.174\end{pmatrix}$$

$$Q=\begin{pmatrix} f(x_1) & f(t_1,x_1) \\ f(s_1,x_1) & f(s_1,t_1,x_1)\end{pmatrix}=\begin{pmatrix} 0.3 & 0.204 \\ 0.108 & 0.0648\end{pmatrix},$$

the eigenvalues of $P^{-1}Q$ and $P'^{-1}Q'$ are 0.533 and 0.109, and the corresponding eigenmatrices are

$$A_1=\begin{pmatrix} 0.196 & -0.625 \\ -0.981 & 0.781\end{pmatrix}, A_2=\begin{pmatrix} 0.514 & -0.287 \\ -0.857 & 0.958\end{pmatrix},$$

for $P^{-1}Q$ and $P'^{-1}Q'$ respectively. Thus, letting $E_i = \mathrm{diag}(a_{1(i)}, a_{2(i)})$ and

$$P_1=\begin{pmatrix} 1.000 & f(s_1|y_1) \\ 1.000 & f(s_1|y_2)\end{pmatrix}, P_2=\begin{pmatrix} 1.000 & f(t_1|y_1) \\ 1.000 & f(t_1|y_2)\end{pmatrix},$$

since we can provide $E_1 = \mathrm{diag}(-0.588, -0.469)$ and $E_2 = \mathrm{diag}(0.257, 0.287)$, noting that $P_1 = E_1 A_1^{-1}$ and $P_2 = E_2 A_2^{-1}$, the followings can be derived:

$$P_1=\begin{pmatrix} 1.000 & 0.800 \\ 1.000 & 0.200\end{pmatrix}, P_2=\begin{pmatrix} 1.000 & 0.300 \\ 1.000 & 0.600\end{pmatrix}.$$

Thus, $M = \mathrm{diag}(f(u_1), f(u_2)) = P_2'^{-1} P P_1^{-1} = \mathrm{diag}(0.45, 0.55)$ and the causal effects of X on Y are $f(y_1|\mathrm{set}(X = x_1)) = 0.533\times 0.45/0.3 = 0.8$ and $f(y_1|\mathrm{set}(X = x_0)) = 1.000 - (1.000 - 0.109)\times 0.55/0.7 = 0.3$, respectively.

Second, we consider the situation where there are bidirected arrows shown in Fig. 3. Under this situation, we can not evaluate the causal effects of X on Y by Theorem 1. However, under the monotonic assumption, we can provide the tightest bounds on the causal effects as

$$0.000 < P(y_1|\mathrm{set}(X=x_0))$$
$$< \min\{0.567, 0.549, 0.384, 0.478\} = 0.384$$

$$\max\{0.205, -0.044, 0.151, 0.338\} = 0.338$$
$$< P(y_1|\text{set}(X = x_0)) < 1.000,$$

based on equations (15) and (16).

6 DISCUSSION

This paper derived the graphical identifiability criteria for causal effects based on causal modeling in observational studies with an unobserved multi-categorical exposure/outcome variable. In addition, when unmeasured variables exist between an unobserved outcome variable and its proxy variables, we provided the tightest bounds on causal effects by using Balke's LP program method. The results of this paper enable us to evaluate causal effects when it is difficult to observe an exposure/outcome variable.

ACKNOWLEGDEMENT

This research was partly supported by the Kurata Foundation, the Mazda Foundation and the Ministry of Education, Culture, Sports, Science and Technology of Japan.

REFERENCES

Balke, A. (1995). Probabilistic Counterfactuals: Semantics, Computation, and Applications, UCLA Cognitive Systems Laboratory, Technical Report.

Balke, A. and Pearl, J. (1997). Bounds on Treatment Effects from Studies with Imperfect Compliance. *Journal of the American Statistical Association* Vol. 92, pp.1171-1176.

Bollen, K. A. (1989). *Structural equations with latent variables*. John Wiley & Sons.

Bross, I. D. J. (1958). How to use ridit analysis. *Biometrics*, **14**, 18-38.

Finch, B. K. (2003). Socioeconomic gradients and low birth-weight: empirical and policy considerations. *Health Services Research*, **38**, 1819-1842.

Fleiss, J. L., Chilton N. W. and Wallenstein S. (1976). Ridit analysis in dental clinical studies. *Journal of Dental Research*, **58**, 2080-2084,

Geiger, D., Verma, T. S. & Pearl, J. (1990). Identifying independence in Bayesian networks. *Networks*, **20**, 507-34.

Goodman, L. A. (1973). The analysis of multidimensional contingency tables when some variables are posterior to others: a modified path analysis approach. *Biometrika*, **60**, 179-92.

Goodman, L. A. (1974a). The analysis of systems of qualitative variables when some of the variables are unobservable: A modified path analysis approach. *American Journal of Sociology*, **79**, 1179-259.

Goodman, L. A. (1974b). Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika*, **61**, 215-31.

Hagenaars, J. A. (1993). *Loglinear models with latent variables*. Sage Publications.

Hoffmann, B and Jockel K. H. (2006). Diesel exhaust and coal mine dust: Lung cancer risk in occupational settings. *Annals of the New York Academy of Sciences*, **1076**, 253-65.

Kuroki, M. (2007). Graphical identifiability criteria for causal effects in studies with an unobserved treatment/response variable. *Biometrika*, accepted.

Kuroki, M., Cai, Z. and Motogaito, H. (2005). Graphical identifiability criteria for total effects by using surrogate variables. *Proceeding of the 21st Conference on Uncertainty in Artificial Intelligence*, 340-345.

Pearl, J. (1988). *Probabilistic reasoning in intelligence systems*. Morgan Kaufmann Publishers.

Pearl, J. (2000). *Causality : models, reasoning, and inference*. Cambridge University Press.

Rosenbaum, P. and Rubin, D. B. (1983). The Central Role of the Propensity Score in Observational Studies for Causal Effects, *Biometrika*, Vol. 70, pp. 41-55.

Shpitser, I. and Pearl, J. (2006). Identification of Joint Interventional Distributions in Recursive Semi-Markovian Causal Models, *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, 1219-1226.

Tian, J. & Pearl, J. (2002). A general identification condition for causal effects. *Proceeding of the 18th National Conference on Artificial Intelligence*, 567-73.

Wold, H. O. (1954). Causality and econometrics. *Econometrika*, **22**, 162-177.

Wright, S. (1923). The theory of path coefficients: a reply to Niles' criticism. *Genetics*, **8**, 239-255.

Wright, S. (1934). The method of path coefficients. *Annals of Mathematical Statistics*, **5**, 161-215.

Partitioned Linear Programming Approximations for MDPs

Branislav Kveton
Intel Research
Santa Clara, CA
branislav.kveton@intel.com

Milos Hauskrecht
Department of Computer Science
University of Pittsburgh
milos@cs.pitt.edu

Abstract

Approximate linear programming (ALP) is an efficient approach to solving large factored Markov decision processes (MDPs). The main idea of the method is to approximate the optimal value function by a set of basis functions and optimize their weights by linear programming (LP). This paper proposes a new ALP approximation. Comparing to the standard ALP formulation, we decompose the constraint space into a set of low-dimensional spaces. This structure allows for solving the new LP efficiently. In particular, the constraints of the LP can be satisfied in a compact form without an exponential dependence on the treewidth of ALP constraints. We study both practical and theoretical aspects of the proposed approach. Moreover, we demonstrate its scale-up potential on an MDP with more than 2^{100} states.

1 Introduction

Markov decision processes (MDPs) [19] are an established framework for solving sequential decision problems under uncertainty. Unfortunately, traditional methods for solving MDPs, such as value and policy iteration, are unsuitable for solving real-world problems. These problems are generally structured, and their state and action spaces are represented by state and action variables. The size of these problems is naturally exponential in the number of the variables, and so are their exact solutions. Approximate linear programming (ALP) [21] has emerged as a promising approach to solving these problems efficiently [6, 12, 15].

The main idea of this method is to approximate the optimal value function by a set of basis functions and optimize their weights by linear programming (LP). The optimization can be performed in a structured manner [10, 20]. The structure is a result of combining the structure of factored MDPs and linear value function approximations.

The complexity of computing exact ALP solutions [10, 20] is exponential in the treewidth of the dependency graph that represents the constraint space in ALP. Therefore, when the treewidth of an ALP is large, its exact solution is infeasible. This type of problems can be still solved approximately using Monte Carlo constraint sampling [7, 14]. This approach can be interpreted as an outer approximation to the feasible region of the ALP.

In this work, we propose inner approximations to the feasible region. In comparison to the standard ALP formulation, the constraint space is factored into a set of subspaces. This structure allows for solving the new LP more efficiently. In particular, its constraints can be satisfied in a compact form without an exponential dependence on the treewidth of the original constraint space. We investigate both practical and theoretical aspects of the approach. In addition, we demonstrate that the approach yields an exponential speedup over ALP.

The paper is organized as follows. First, we review factored MDPs [5] and linear value function approximations [2, 22]. Second, we discuss in detail existing work on approximate linear programming. Third, we propose a novel partitioned ALP formulation and study its properties. Finally, we evaluate the quality of the approximation on decision problems with more than 2^{100} states.

2 Factored MDPs

Many real-world decision problems are naturally described in a factored form. Factored MDPs [5] allow for a compact representation of this structure.

A *factored MDP* [5] is a 4-tuple $\mathcal{M} = (\mathbf{X}, \mathcal{A}, P, R)$, where $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a state space represented by a set of state variables, $\mathcal{A} = \{a_1, \ldots, a_m\}$ is a finite set of actions[1], $P(\mathbf{X}' \mid \mathbf{X}, \mathcal{A})$ is a transition function, which represents the dynamics of the MDP, and R is a reward function assigning immediate payoffs to state-action configurations. The state of the system is completely observed and given by a vector of value assignments $\mathbf{x} = (x_1, \ldots, x_n)$.

[1] For simplicity of exposition, we consider an MDP model with a single action variable $\mathcal{A}$. Our ideas straightforwardly generalize to MDPs with factored action spaces [11].

Transition model: The transition model is represented by a conditional probability distribution $P(\mathbf{X}' \mid \mathbf{X}, \mathcal{A})$, where $\mathbf{X}$ and $\mathbf{X}'$ denote the state variables at two successive time steps. Since the full tabular representation of $P(\mathbf{X}' \mid \mathbf{X}, \mathcal{A})$ is infeasible when the number of state variables is large, we assume that the distribution factors along $\mathbf{X}'$ as:

$$P(\mathbf{X}' \mid \mathbf{X}, a) = \prod_{i=1}^{n} P(X_i' \mid \mathsf{Par}(X_i'), a) \tag{1}$$

and is described compactly by a *dynamic Bayesian network (DBN)* [8]. The network reflects independencies among the variables $\mathbf{X}$ and $\mathbf{X}'$ given an action a. One-step dynamics of every state variable is given by its conditional probability distribution $P(X_i' \mid \mathsf{Par}(X_i'), a)$, where $\mathsf{Par}(X_i') \subseteq \mathbf{X}$ is the parent set of X_i'. The parent set is usually a small subset of state variables which simplifies the parameterization of the model.

Reward model: The reward model is factored similarly to the transition model. Specifically, the reward function:

$$R(\mathbf{x}, a) = \sum_{j} R_j(\mathbf{x}_j, a) \tag{2}$$

is an additive function of local reward functions defined on the subsets $\mathbf{X}_j$ and $\mathcal{A}$. These local functions are compactly represented by reward nodes R_j, which are conditioned on their parent sets $\mathsf{Par}(R_j) = \mathbf{X}_j \cup \mathcal{A}$.

Optimal value function and policy: The quality of a policy π is measured by the *infinite horizon discounted reward* $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where $\gamma \in [0, 1)$ is a *discount factor* and r_t is the immediate reward at the time step t. In such a setting, there always exists an *optimal policy* π^* which is stationary and deterministic [19]. The policy is greedy with respect to the *optimal value function* V^*, which is a fixed point of the Bellman equation [1]:

$$V^*(\mathbf{x}) = \max_{a} \left[R(\mathbf{x}, a) + \gamma \mathbb{E}_{P(\mathbf{x}' \mid \mathbf{x}, a)}[V^*(\mathbf{x}')] \right]. \tag{3}$$

Similarly to the above equation, all expectation terms in the rest of the paper are written compactly as $\mathbb{E}_{P(\mathbf{x})}[f(\mathbf{x})]$.

3 Solving factored MDPs

Markov decision processes can be solved by exact dynamic programming (DP) methods in polynomial time in the size of their state space [19]. Unfortunately, the space space $\mathbf{X}$ of factored MDPs is exponential in the number of state variables. Therefore, the DP methods are unsuitable for solving these problems. Since a factored representation of an MDP does not guarantee a structure in its solution [13], we resort to value function approximations.

In this work, we focus on the *linear value function approximation* [2, 22]:

$$V^{\mathbf{w}}(\mathbf{x}) = \sum_{i} w_i f_i(\mathbf{x}). \tag{4}$$

The approximation restricts the form of the value function to the linear combination of basis functions $f_i(\mathbf{x})$, where $\mathbf{w}$ is a vector of optimized weights. The basis functions $f_i(\mathbf{x})$ are arbitrary functions, which are usually restricted to small subsets of state variables $\mathbf{X}_i$ [2, 13]. The functions play the same role as features in machine learning. They are usually provided by domain experts but can also be discovered automatically [18, 16].

4 Approximate linear programming

Various techniques for optimizing the linear value function approximation have been studied and analyzed [3]. We focus on *approximate linear programming (ALP)* [21], which restates this problem as a linear program:

$$\begin{aligned} \text{minimize}_{\mathbf{w}} \quad & \sum_{i} w_i \alpha_i \\ \text{subject to:} \quad & \sum_{i} w_i F_i(\mathbf{x}, a) - R(\mathbf{x}, a) \geq 0 \\ & \forall\, \mathbf{x} \in \mathbf{X}, a \in \mathcal{A}; \end{aligned} \tag{5}$$

where $\mathbf{w}$ denotes the variables in the LP, α_i is a *basis function relevance weight*:

$$\alpha_i = \mathbb{E}_{\psi(\mathbf{x})}[f_i(\mathbf{x})], \tag{6}$$

$\psi(\mathbf{x}) \geq 0$ is a *state relevance density function* that weights the quality of the approximation, and:

$$F_i(\mathbf{x}, a) = f_i(\mathbf{x}) - \gamma \mathbb{E}_{P(\mathbf{x}' \mid \mathbf{x}, a)}[f_i(\mathbf{x}')] \tag{7}$$

denotes the difference between the basis function $f_i(\mathbf{x})$ and its discounted *backprojection*. This linear program is feasible if the set of basis functions includes a constant function $f_0(\mathbf{x}) \equiv 1$. We assume that such a basis function is present.

Since our basis functions $f_i(\mathbf{x})$ are often restricted to small subsets of state variables, expectation terms in the ALP formulation (5) can be computed efficiently [10]. For instance, the backprojection terms can be rewritten as:

$$\mathbb{E}_{P(\mathbf{x}' \mid \mathbf{x}, a)}[f_i(\mathbf{x}')] = \mathbb{E}_{P(\mathbf{x}_i' \mid \mathbf{x}, a)}[f_i(\mathbf{x}_i')], \tag{8}$$

where $\mathbf{X}_i'$ is a lower dimensional state space corresponding to the basis function $f_i(\mathbf{x})$, and $P(\mathbf{x}_i' \mid \mathbf{x}, a)$ is a distribution defined on this subspace. Similarly, state relevance weights α_i can be computed efficiently if the state relevance density $\psi(\mathbf{x})$ is structured.

4.1 Solving ALP formulations

The major problem in solving ALP formulations efficiently is in satisfying their constraints. This problem is hard since the number of the constraints is exponential in the number of state variables. Fortunately, the constraints exhibit some structure. The structure is a result of combining linear value function approximations (Equation 4) with factored reward and transition models (Equations 1 and 2). Therefore, ALP

constraints can be satisfied in a structured form and without being enumerated exhaustively.

Based on these observations, Guestrin *et al.* [10] proposed a variable elimination method [9] that rewrites the constraint space compactly. Schuurmans and Patrascu [20] solved the constraint satisfaction problem by the cutting plane method [4]. The approach iteratively searches for the most violated constraint:

$$\arg\min_{\mathbf{x},a} \left[\sum_i w_i^{(t)} F_i(\mathbf{x}, a) - R(\mathbf{x}, a) \right] \tag{9}$$

with respect to the solution $\mathbf{w}^{(t)}$ of a relaxed ALP. The most violated constraint is added to the linear program, which is in turn resolved for a new vector $\mathbf{w}^{(t+1)}$. This procedure is iterated until no violated constraint is found. In such a case, the vector $\mathbf{w}^{(t)}$ is an optimal solution to the ALP.

The space complexity of both constraint satisfaction methods [10, 20] is exponential in the treewidth of the constraint space. As a result, the methods are unsuitable for problems with a large treewidth. However, such problems can be still solved approximately. For instance, de Farias and Van Roy [7] proposed Monte Carlo approximations of the constraint space. Kveton and Hauskrecht [14] showed how to search for the most violated constraint (Equation 9) using Markov chain Monte Carlo (MCMC) sampling.

4.2 Theoretical analysis

The quality of the ALP formulation has been studied by de Farias and Van Roy [6]. Based on their work, we conclude that ALP minimizes the $\mathcal{L}_1$-norm error $\|V^* - V^{\mathbf{w}}\|_{1,\psi}$. The following theorem draws a parallel between optimizing this objective and the max-norm error $\|V^* - V^{\mathbf{w}}\|_\infty$.

Theorem 1 (de Farias and Van Roy [6])**.** *Let $\widetilde{\mathbf{w}}$ be a solution to the ALP formulation (5). Then the expected error of the value function $V^{\widetilde{\mathbf{w}}}$ can be bounded as:*

$$\left\| V^* - V^{\widetilde{\mathbf{w}}} \right\|_{1,\psi} \leq \frac{2}{1-\gamma} \min_{\mathbf{w}} \|V^* - V^{\mathbf{w}}\|_\infty ,$$

where $\|\cdot\|_{1,\psi}$ is an $\mathcal{L}_1$-norm weighted by the state relevance density function ψ and $\|\cdot\|_\infty$ is the max-norm.

De Farias and Van Roy [6] also proved a tighter version of Theorem 1, which reweights the error $\|V^* - V^{\mathbf{w}}\|_\infty$.

5 Partitioned ALP

In this section, we propose a novel approximate linear programming formulation. In comparison to the standard ALP (5), the proposed formulation has an additional structure in its constraint space. The structure allows for controlling the complexity of solving the new LP.

The LP solves a more restrictive problem than the standard ALP. As a result, the formulation can be viewed as an inner approximation to the feasible region of the ALP (Figure 1). This differentiates our work from existing ALP approximations [7, 14]. These approximations are based on constraint sampling. As a result, they approximate the feasible region of the ALP from outside.

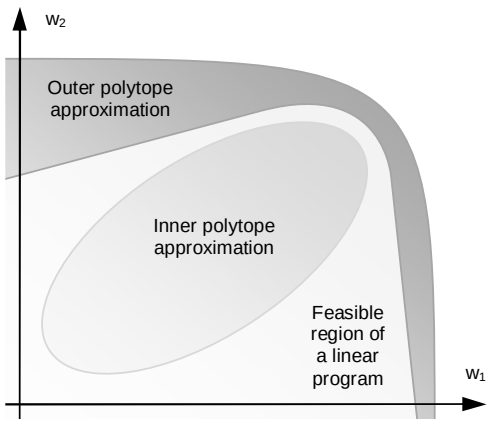


Figure 1: An illustration of inner and outer approximations to the feasible region of a linear program.

5.1 An illustrative example

First, let us consider an optimization problem:

$$\begin{aligned} \text{minimize}_{\mathbf{w},h} \quad & w_1\alpha_1 + w_2\alpha_2 + h \\ \text{subject to:} \quad & w_1 F_1(x_1) + w_2 F_2(x_2) + h \geq 0 \\ & \forall\, x_1 \in X_1, x_2 \in X_2; \end{aligned} \tag{10}$$

where $\mathbf{w} = (w_1, w_2)$ denotes the main optimized variables, and h is an auxiliary variable that guarantees the feasibility of the LP. This problem involves $|X_1 \times X_2| = |X_1| \times |X_2|$ constraints. If the number of the constraints is large, a suboptimal but feasible solution to the problem can be obtained by solving a new linear program:

$$\begin{aligned} \text{minimize}_{\mathbf{w},h} \quad & w_1\alpha_1 + w_2\alpha_2 + h \\ \text{subject to:} \quad & h_1 + h_2 = h \\ & w_1 F_1(x_1) + h_1 \geq 0 \quad \forall\, x_1 \in X_1 \\ & w_2 F_2(x_2) + h_2 \geq 0 \quad \forall\, x_2 \in X_2; \end{aligned} \tag{11}$$

where h_1 and h_2 are new auxiliary variables that guarantee the feasibility of the LP. Note that the new LP decomposes the original constraint $w_1 F_1(x_1) + w_2 F_2(x_2) + h \geq 0$ into two smaller constraint spaces with $|X_1| + |X_2|$ constraints. Therefore, it is typically faster to solve the new LP than our original problem (10). In the next section, we show how to apply similar ideas in the context of ALP.

5.2 Partitioned ALP formulation

Similarly to Section 5.1, we may decompose the constraint space in the ALP formulation (5). Formally, the *partitioned*

ALP (PALP) formulation with K constraint spaces is given by a linear program:

$$\begin{aligned} \text{minimize}_{\mathbf{w}} \quad & \sum_i w_i \alpha_i \qquad (12) \\ \text{subject to:} \quad & \mathbf{D}\mathbf{M}_{\mathbf{w}}(\mathbf{x}, a)^\intercal \geq 0 \quad \forall\, \mathbf{x} \in \mathbf{X}, a \in \mathcal{A}; \end{aligned}$$

where:

$$\mathbf{M}_{\mathbf{w}}(\mathbf{x}, a) = (w_1 F_1(\mathbf{x}, a), \ldots, -R_1(\mathbf{x}_1, a), \ldots) \qquad (13)$$

is a vector whose i-th element corresponds to the i-th term in the ALP constraint, and the *partitioning matrix*:

$$\mathbf{D} = \begin{pmatrix} d_{1,1} & d_{1,2} & d_{1,3} & \cdots \\ d_{2,1} & d_{2,2} & d_{2,3} & \cdots \\ d_{3,1} & d_{3,2} & d_{3,3} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \qquad (14)$$

determines how the ALP constraint decomposes into the K new constraint spaces. Specifically, the term $d_{k,i}$ measures the contribution of the i-th term in the ALP constraint to the k-th constraint space. Due to this interpretation, we assume that all terms $d_{k,i}$ are non-negative and that the partitioning matrix $\mathbf{D}$ is normalized such that the equality $\sum_k d_{k,i} = 1$ holds for all i. Under such assumptions, it is trivial to show that the satisfaction of the K constraints $\mathbf{D}\mathbf{M}_{\mathbf{w}}(\mathbf{x}, a)^\intercal \geq 0$ leads to the satisfaction of a corresponding ALP constraint. The claim can be proved based on the identity:

$$\mathbf{1}\mathbf{D}\mathbf{M}_{\mathbf{w}}(\mathbf{x}, a)^\intercal = \sum_i w_i F_i(\mathbf{x}, a) - R(\mathbf{x}, a), \qquad (15)$$

where $\mathbf{1}$ is a row vector of ones. It follows that every PALP solution is feasible in a corresponding ALP.

Similarly to ALP, the feasibility of the PALP formulation is guaranteed if the set of basis functions includes a constant function $f_0(\mathbf{x}) \equiv 1$. We assume that the function is present in all K constraint spaces. In each of them, we define a new weight w_0^k, which reflects the contribution of this function. As a result of these changes, the PALP formulation slightly changes its form:

$$\begin{aligned} \text{minimize}_{\mathbf{w}} \quad & \sum_i w_i \alpha_i + w_0 \qquad (16) \\ \text{subject to:} \quad & \sum_k w_0^k = w_0 \\ & \mathbf{D}\mathbf{M}_{\mathbf{w}}(\mathbf{x}, a)^\intercal + (1-\gamma)(w_0^1, \ldots, w_0^K)^\intercal \geq 0 \\ & \forall\, \mathbf{x} \in \mathbf{X}, a \in \mathcal{A}. \end{aligned}$$

In the rest of the paper, we use the above and original PALP formulations interchangeably.

5.3 Partitioning matrix

The partitioning matrix $\mathbf{D}$ allows for trading off the quality and complexity of PALP solutions. To achieve high-quality and tractable approximations, the rows of the matrix should reflect tree decompositions of the *cost network* corresponding to ALP constraints (Figure 2). The width of the decompositions should be small since the complexity of satisfying a single constraint space is exponential in its treewidth [10].

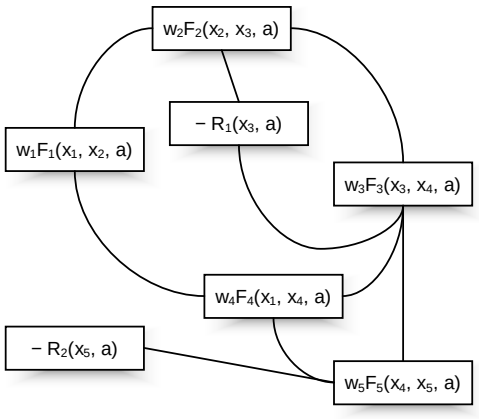


Figure 2: A graphical representation of a cost network. The rectangular nodes represent functions, which are defined on some subset of variables. Two nodes in the cost network are connected if their functions share at least one variable.

How to generate the best PALP approximation within a certain complexity limit is an open question. In the experimental section, we build the matrix $\mathbf{D}$ based on a heuristic. The heuristic generates a constraint space for every expectation term $F_k(\mathbf{x}, a)$ in Equation 9. This constraint space consists of the term $w_k F_k(\mathbf{x}, a)$ and its cost network neighbors. The constraint space is not included in the matrix $\mathbf{D}$ if its terms constitute a subset of another constraint space.

This decomposition of our initial problem can be viewed as optimizing K smaller MDPs, which have overlapping state and action spaces, and share value functions. To clarify the construction of the matrix $\mathbf{D}$, we demonstrate it on the cost network in Figure 2. The cost network involves 7 functions, out of which 5 have the form of $w_k F_k(\mathbf{x}, a)$. Therefore, the corresponding matrix $\mathbf{D}$ has 5 rows and 7 columns:

$$\mathbf{D} = \begin{pmatrix} 0.3\bar{3} & 0.3\bar{3} & 0 & 0.25 & 0 & 0 & 0 \\ 0.3\bar{3} & 0.3\bar{3} & 0.25 & 0 & 0 & 0.5 & 0 \\ 0 & 0.3\bar{3} & 0.25 & 0.25 & 0.3\bar{3} & 0.5 & 0 \\ 0.3\bar{3} & 0 & 0.25 & 0.25 & 0.3\bar{3} & 0 & 0 \\ 0 & 0 & 0.25 & 0.25 & 0.3\bar{3} & 0 & 1 \end{pmatrix}. \qquad (17)$$

Non-zero entries $d_{k,i}$ in the matrix indicate that the i-th cost network term is present in the k-th constraint space.

5.4 Solving PALP formulations

The PALP formulation (12) is similar to the ALP formulation (5). As a result, it can be solved in a similar fashion. In the experimental section, we implemented the cutting plane method for solving linear programs (Figure 3). In principle, any method for solving ALPs (Section 4.1) can be adapted to PALPs.

Inputs:
a factored MDP $\mathcal{M} = (\mathbf{X}, \mathcal{A}, P, R)$
basis functions $f_0(\mathbf{x}), f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots$
initial basis function weights $\mathbf{w}^{(0)}$
a separation oracle $\mathcal{O}$

Algorithm:
initialize a relaxed PALP formulation
$t = 0$
while a stopping criterion is not met
  for every constraint space $k = 1, \ldots, K$
    query the oracle $\mathcal{O}$ for a violated constraint $(\mathbf{x}_{\mathcal{O}}, a_{\mathcal{O}})$
    if the constraint $(\mathbf{x}_{\mathcal{O}}, a_{\mathcal{O}})$ is violated
      add the constraint to the relaxed PALP
    resolve the LP for a new vector $\mathbf{w}^{(t+1)}$
    $t = t + 1$

Outputs:
basis function weights $\mathbf{w}^{(t)}$

Figure 3: Pseudo-code implementation of the cutting plane method for solving PALP formulations.

5.5 Theoretical analysis

In this section, we discuss the quality of the PALP formulation (12). First, we prove that its solution is an upper bound on the optimal value function V^*.

Proposition 1. *Let $\widetilde{\mathbf{w}}$ be a solution to the PALP formulation (12). Then $V^{\widetilde{\mathbf{w}}} \geq V^*$.*

Proof: Since $\widetilde{\mathbf{w}}$ is a solution to the PALP formulation (12), it is also a suboptimal solution to the ALP formulation (5). Therefore, the constraint $V^{\widetilde{\mathbf{w}}} \geq \mathcal{T}^* V^{\widetilde{\mathbf{w}}}$ is satisfied. Furthermore, note that the Bellman operator $\mathcal{T}^*$ is both monotonic and contracting. Hence, the inequality $V^{\widetilde{\mathbf{w}}} \geq \mathcal{T}^* V^{\widetilde{\mathbf{w}}}$ yields the following sequence of inequalities:

$$V^{\widetilde{\mathbf{w}}} \geq \mathcal{T}^* V^{\widetilde{\mathbf{w}}} \geq \mathcal{T}^* \mathcal{T}^* V^{\widetilde{\mathbf{w}}} \geq \cdots \geq V^*.$$

This step concludes our proof. ■

The above result allows us to restate the objective $\mathbb{E}_{\psi}[V^{\mathbf{w}}]$ in PALP.

Proposition 2. *The objective in the PALP formulation (12) can be rewritten as $\|V^* - V^{\mathbf{w}}\|_{1,\psi}$, where $\|\cdot\|_{1,\psi}$ is an $\mathcal{L}_1$-norm weighted by the state relevance density function ψ.*

Proof: Follows from the fact that all solutions to the PALP formulation (12) satisfy the constraint $V^{\mathbf{w}} \geq V^*$. ■

Based on Proposition 2, we conclude that PALP optimizes the linear value function approximation with respect to the reweighted $\mathcal{L}_1$-norm error $\|V^* - V^{\mathbf{w}}\|_{1,\psi}$. The following theorem draws a parallel between optimizing this objective and the max-norm error $\|V^* - V^{\mathbf{w}}\|_{\infty}$.

Theorem 2. *Let $\widetilde{\mathbf{w}}$ be a solution to the PALP formulation (12). Then the expected error of the value function $V^{\widetilde{\mathbf{w}}}$ can be bounded as:*

$$\left\|V^* - V^{\widetilde{\mathbf{w}}}\right\|_{1,\psi} \leq \frac{2}{1-\gamma} \min_{\mathbf{w}} \|V^* - V^{\mathbf{w}}\|_{\infty} + \frac{K\delta}{1-\gamma},$$

where $\|\cdot\|_{1,\psi}$ is an $\mathcal{L}_1$-norm weighted by the state relevance density function ψ, $\|\cdot\|_{\infty}$ is the max-norm, δ is a scalar that reflects how hard is to make an ALP solution feasible in our PALP formulation, and K denotes the number of constraint spaces in the PALP.

Proof: Our proof is similar to the proof of Theorem 2 by de Farias and Van Roy [6]. The vectors $\widetilde{\mathbf{w}}$, $\overline{\mathbf{w}}$, and $\mathbf{w}^*$ denote an optimal solution to the PALP formulation, its suboptimal solution, and the vector that minimizes the max-norm error $\|V^* - V^{\mathbf{w}}\|_{\infty}$, respectively. First, we bound the objective in the PALP as follows:

$$\begin{aligned}\left\|V^* - V^{\widetilde{\mathbf{w}}}\right\|_{1,\psi} &\leq \left\|V^* - V^{\overline{\mathbf{w}}}\right\|_{1,\psi} \\ &\leq \left\|V^* - V^{\overline{\mathbf{w}}}\right\|_{\infty}.\end{aligned}$$

Second, we bound the max-norm error of $V^{\overline{\mathbf{w}}}$ by the triangle inequality:

$$\left\|V^* - V^{\overline{\mathbf{w}}}\right\|_{\infty} \leq \left\|V^* - V^{\widehat{\mathbf{w}}}\right\|_{\infty} + \left\|V^{\widehat{\mathbf{w}}} - V^{\overline{\mathbf{w}}}\right\|_{\infty},$$

where $\widehat{\mathbf{w}}$ is an arbitrary solution to the ALP formulation. In the rest of the proof, we bound the two terms on the right-hand side of the inequality. The first term reflects how hard is to fit the linear value function approximation to the value function V^*. If the vector $\widehat{\mathbf{w}}$ is set such that:

$$\widehat{\mathbf{w}} = \mathbf{w}^* + \frac{1+\gamma}{1-\gamma} \left\|V^* - V^{\mathbf{w}^*}\right\|_{\infty} i_0,$$

where $i_0 = (1, 0, \ldots, 0)$ is an indicator of the constant basis function $f_0(\mathbf{x}) \equiv 1$, the following inequality:

$$\left\|V^* - V^{\widehat{\mathbf{w}}}\right\|_{\infty} \leq \frac{2}{1-\gamma} \left\|V^* - V^{\mathbf{w}^*}\right\|_{\infty}$$

holds [6]. The second term reflects how hard it to make the ALP solution $\widehat{\mathbf{w}}$ feasible in the PALP. If the vector $\overline{\mathbf{w}}$ is set such that:

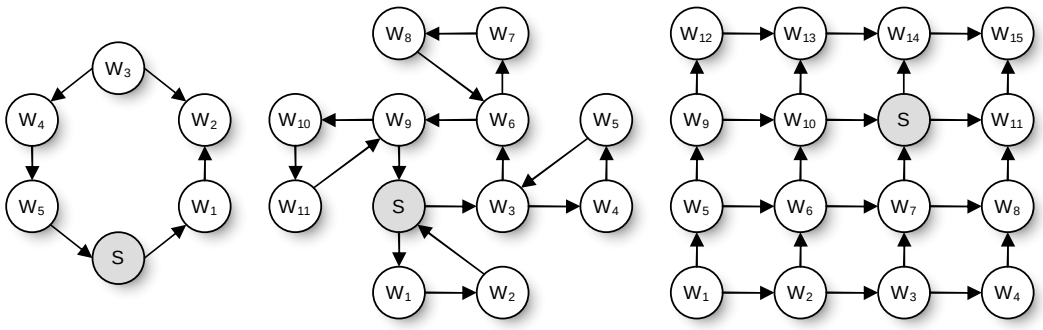
$$\overline{\mathbf{w}} = \widehat{\mathbf{w}} + \frac{K\delta}{1-\gamma} i_0,$$

where $i_0 = (1, 0, \ldots, 0)$ is an indicator of the constant basis function $f_0(\mathbf{x}) \equiv 1$, $\delta = -\min_{\mathbf{x},a} \min(\mathbf{DM}_{\widehat{\mathbf{w}}}(\mathbf{x}, a)^{\intercal})$, and the function $\min(\mathbf{DM}_{\widehat{\mathbf{w}}}(\mathbf{x}, a)^{\intercal})$ computes the minimum of the vector $\mathbf{DM}_{\widehat{\mathbf{w}}}(\mathbf{x}, a)^{\intercal}$, we can guarantee the feasibility of $\overline{\mathbf{w}}$. The proof is based on the observation that all constraints in the feasible PALP formulation (16) are satisfied when the weights $\overline{w}_0^k$ are set such that:

$$\overline{w}_0^k = \frac{1}{K} \widehat{w}_0 + \frac{\delta}{1-\gamma}.$$

Based on this setting, the max-norm error between $V^{\widehat{\mathbf{w}}}$ and $V^{\overline{\mathbf{w}}}$ is bounded as:

$$\left\|V^{\widehat{\mathbf{w}}} - V^{\overline{\mathbf{w}}}\right\|_{\infty} \leq \frac{K\delta}{1-\gamma}.$$



(a) **(b)** **(c)**

Figure 4: An illustration of three network administration topologies: **a.** 6-ring, **b.** 12-ring-of-rings, and **c.** 4×4 grid. The gray and white nodes represent the server and workstations, respectively. The computers are connected along the arrows.

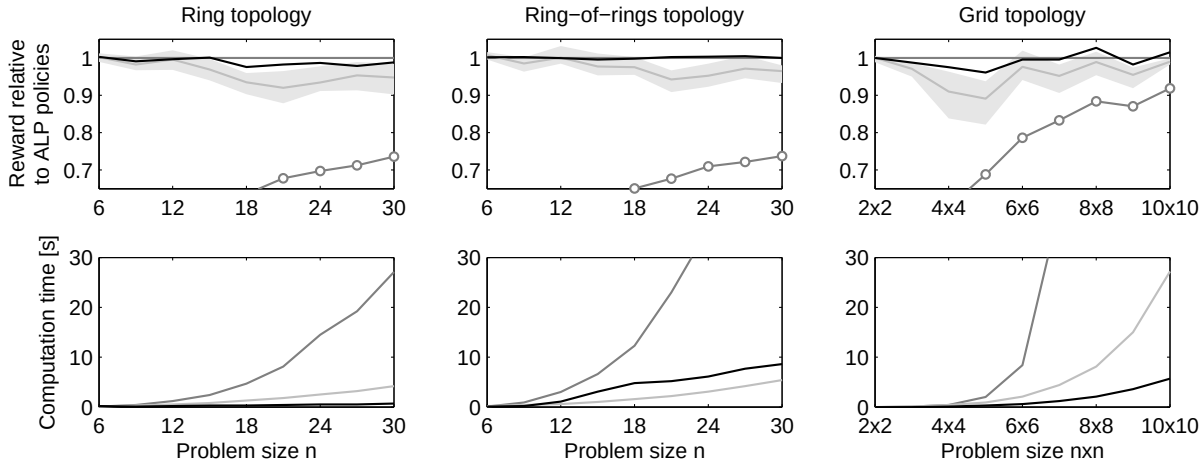


Figure 5: Comparison of four policies for solving the network administration problem. The first policy is obtained by PALP (black lines), the second one by ALP (dark gray lines), the third one by ALP with randomly sampled constraints (light gray lines), and the fourth policy is the server heuristic (dark gray lines with circles). The policies are compared by their reward, which is measured relatively to the reward of ALP policies, and computation time (in seconds). The variance in the rewards of sampled ALP approximations is depicted by gray areas. All results are reported as functions of increasing problem sizes (n).

This step concludes our proof. ■

The above result can be interpreted as follows. PALP yields a close approximation $V^{\widetilde{\mathbf{w}}}$ to the optimal value function V^* if the function V^* lies in the span of basis functions and the penalty δ for partitioning the ALP constraint space is small. Unfortunately, we do not have a good bound for the penalty term δ. The value of δ can be as bad as $\|\widehat{\mathbf{w}}\|_1 + R_{\max}$, where $R_{\max}$ denotes the maximum immediate reward in an MDP. Hence, the bound in Theorem 2 is not very tight in practice. Nevertheless, it provides valuable insights into two sources of errors for PALP approximations.

6 Experiments

The objective of the experimental section is to demonstrate the quality and scale-up potential of PALP approximations. The approximations are studied with respect to ALP, which is a state-of-the-art approach to solving large-scale factored MDPs. Our experiments are performed on various forms of the network administration problem [10]. This is a standard benchmark for testing the scalability of MDP algorithms.

6.1 Experimental setup

The network administration problem involves a network of randomly crashing computers. When a computer crashes, it increases the probability of its network neighbors crashing. The objective is to reboot crashed computers to restore their functionality and prevent further spreading of their failures into the network. Examples of three network topologies are shown in Figure 4. Each network consists of one server and several workstations. The difference between the two types of the computers is in the reward for keeping them running.

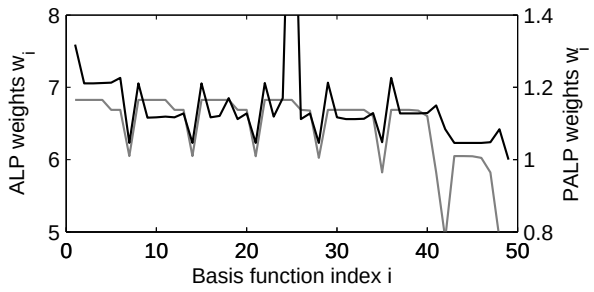


Figure 6: Basis function weights w_i obtained by ALP (dark gray line) and PALP (black line) on the 7×7 grid network administration problem.

The immediate reward for keeping a workstation running is 1. The reward for keeping the server running is 2.

The network administration problem is a challenging MDP due to the size of its state space. Specifically, since the state of the network is a product of individual computer states, it is exponential in the number of computers. Therefore, only small instances of the problem can be solved exactly. In the rest of the section, we focus on large-scale problems and try to solve them through linear value function approximations (Equation 4). In all experiments, we define a basis function $f_i(\mathbf{x}) = x_i$ for every computer X_i. Furthermore, in the ring and ring-of-rings topologies (Figures 4a and 4b), we assign a pairwise basis function $f_{i \to j}(\mathbf{x}) = x_i x_j$ to every network connection $X_i \to X_j$.

Our linear value function approximations are optimized using ALP and PALP formulations. The cutting plane method is employed to solve these LPs exactly and efficiently (Figure 3). In addition, we experiment with ALP formulations, which are solved approximately by Monte Carlo constraint sampling [7]. The number of sampled constraints is $100n$, where n is the number of state variables $\mathbf{X}$. Therefore, it is proportional to the size of solved problems. To demonstrate the non-triviality of learned policies, we also report results of a heuristic for solving our problem. The heuristic places the administrator at the server so the computer is protected from crashing.

6.2 Experimental results

Our main experimental results are summarized in Figure 5. Based on these results, we conclude that PALP policies are almost as good as ALP policies. Specifically, note that the rewards of the policies are within 95 percent of our baseline in all experiments. Unfortunately, these good results cannot be explained by Theorem 2 because our bound is too loose. To explain our results, we tried to investigate the similarity of basis function weights $\mathbf{w}$ obtained by ALP and PALP. As illustrated in Figure 6, the magnitudes of the weights can be very different. However, the weights exhibit similar trends. In turn, value function approximations corresponding to the weights must have similar shapes, and their greedy policies

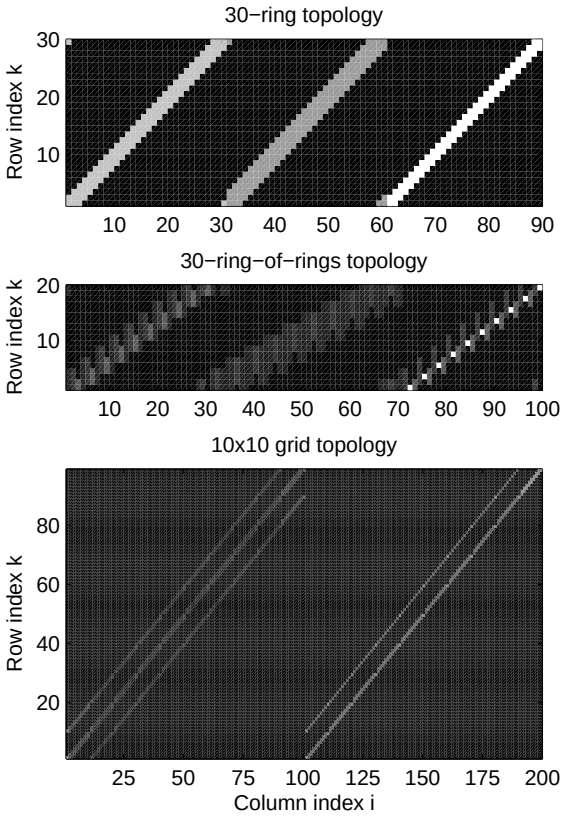


Figure 7: Three partitioning matrices $\mathbf{D}$ corresponding to the network administration problem. The brighter the color of a pixel, the higher the value of the partitioning coefficient $d_{k,i}$. Black pixels represent zero coefficients.

are similar as a result.

Figure 5 also suggests that PALP policies can be computed significantly faster than ALP policies. This speedup results from working with sparse decompositions (Figure 7) of the original constraint space rather than the space itself. Moreover, note that the treewidth of the $n \times n$ network administration problem (Figure 4c) is n. Therefore, the complexity of learning ALP policies for this problem is naturally exponential in n. On the other hand, the complexity of learning PALP policies is polynomial in n. This claim follows from the observation that the number of PALP constraint spaces is n^2 and their treewidth is not dependent on n. As a result, PALP on the grid network provides an exponential speedup over ALP. This result can be verified by the analysis of the computation time trends in Figure 5.

Finally, Figure 5 illustrates that PALP policies are superior to ALP policies, which are obtained by ALP with randomly sampled constraints. In most cases, the PALP policies yield significantly higher rewards than the average sampled ALP approximation. For all larger network administration problems, the policies are as good or better than the best of these

approximations. At the same time, the computation time of the PALP policies is shorter or comparable to the computation time of the sampled approximations.

7 Conclusions

Development of scalable algorithms for solving real-world MDPs is a challenging task. In this work, we investigated a novel approach to approximate linear programming. Comparing to the standard ALP formulation, we decompose the constraint space into a set of low-dimensional spaces. This structure allows for solving the new LP more efficiently. In particular, its constraints can be satisfied in a compact form without an exponential dependence on the treewidth of the original constraint space. Our experiments demonstrate the superiority of the new approach when compared to existing exact and approximate solutions to ALP.

Results of this paper can be extended in several ways. First, we have not addressed the topic of learning good partitioning matrices $\mathbf{D}$. This topic is in many aspects similar to the problem of efficient inference in Bayesian networks. In this context, Meila [17] proposed using a mixture of trees to approximate an arbitrary joint probability distribution defined by a Bayesian network. Second, the bound in Theorem 2 is definitely loose in practice. How to make this bound tight is an interesting open question. Finally, PALP and its benefits should be studied on a more realistic problem than the one presented in the experimental section.

Acknowledgment

We thank anonymous reviewers for helpful comments that led to the improvement of this paper. We also thank Carlos Guestrin for encouragement and positioning this paper in a broader context.

References

[1] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[2] Richard Bellman, Robert Kalaba, and Bella Kotkin. Polynomial approximation – a new computational technique in dynamic programming: Allocation processes. *Mathematics of Computation*, 17(82):155–161, 1963.

[3] Dimitri Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[4] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Belmont, MA, 1997.

[5] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1104–1111, 1995.

[6] Daniela Pucci de Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–856, 2003.

[7] Daniela Pucci de Farias and Benjamin Van Roy. On constraint sampling for the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478, 2004.

[8] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150, 1989.

[9] Rina Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 211–219, 1996.

[10] Carlos Guestrin, Daphne Koller, and Ronald Parr. Max-norm projections for factored MDPs. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 673–682, 2001.

[11] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 14*, pages 1523–1530, 2002.

[12] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[13] Daphne Koller and Ronald Parr. Computing factored value functions for policies in structured MDPs. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 1332–1339, 1999.

[14] Branislav Kveton and Milos Hauskrecht. An MCMC approach to solving hybrid factored MDPs. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1346–1351, 2005.

[15] Branislav Kveton, Milos Hauskrecht, and Carlos Guestrin. Solving factored MDPs with hybrid state and action variables. *Journal of Artificial Intelligence Research*, 27:153–201, 2006.

[16] Sridhar Mahadevan. Samuel meets Amarel: Automating value function approximation using global state space analysis. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 1000–1005, 2005.

[17] Marina Meila. *Learning with Mixtures of Trees*. PhD thesis, Massachusetts Institute of Technology, 1999.

[18] Relu Patrascu, Pascal Poupart, Dale Schuurmans, Craig Boutilier, and Carlos Guestrin. Greedy linear value-approximation for factored Markov decision processes. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 285–291, 2002.

[19] Martin Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York, NY, 1994.

[20] Dale Schuurmans and Relu Patrascu. Direct value-approximation for factored MDPs. In *Advances in Neural Information Processing Systems 14*, pages 1579–1586, 2002.

[21] Paul Schweitzer and Abraham Seidmann. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110:568–582, 1985.

[22] Benjamin Van Roy. *Planning Under Uncertainty in Complex Structured Environments*. PhD thesis, Massachusetts Institute of Technology, 1998.

The Computational Complexity of Sensitivity Analysis and Parameter Tuning

Johan Kwisthout and Linda C. van der Gaag
Department of Information and Computing Sciences, University of Utrecht
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
{johank,linda}@cs.uu.nl

Abstract

While known algorithms for sensitivity analysis and parameter tuning in probabilistic networks have a running time that is exponential in the size of the network, the exact computational complexity of these problems has not been established as yet. In this paper we study several variants of the tuning problem and show that these problems are $\mathsf{NP}^{\mathsf{PP}}$-complete in general. We further show that the problems remain NP-complete or PP-complete, for a number of restricted variants. These complexity results provide insight in whether or not recent achievements in sensitivity analysis and tuning can be extended to more general, practicable methods.

1 Introduction

The sensitivity of the output of a probabilistic network to small changes in the network's parameters, has been studied by various researchers [1, 14, 7, 2, 19, 4]. Whether the parameter probabilities of a network are assessed by domain experts or estimated from data, they inevitably include some inaccuracies. In a sensitivity analysis of the network, the parameter probabilities are varied within a plausible range and the effect of the variation is studied on the output computed from the network, be it a posterior probability or the most likely value of an output variable.

The results of a sensitivity analysis are used, for example, to establish the robustness of the network's output. The results are used also upon engineering a probabilistic network, for example to distinguish between parameters which allow some imprecision and parameters which should be determined as accurately as possible [6]. Another use is for carefully *tuning* the parameter probabilities of a network to arrive at some desired model behavior [3].

Research efforts in sensitivity analysis and parameter tuning for probabilistic networks have resulted in a variety of fundamental insights and computational methods. While the majority of these insights and methods pertain to a one-way sensitivity analysis in which the effect of varying a single parameter probability on a single output probability or output value is studied, recently there also has been some pioneering work on extending these insights to higher-order analyses [2, 6].

The currently available algorithms for sensitivity analysis and parameter tuning of probabilistic networks have a running time that is exponential in the size of a network. This observation suggests that these problems are intractable in general. The actual computational complexity of the problems has not been studied yet, however. In this paper we define several variants of the tuning problem for probabilistic networks and show that these variants are $\mathsf{NP}^{\mathsf{PP}}$-complete in general. We further show that the tuning problem remains NP-comlete, even if the topological structure of the network under study is restricted to a polytree, and PP-complete, even if the number of conditional probability tables involved is bounded.

Given the unfavorable complexity results obtained, even for restricted cases of the tuning problem, we have that we cannot expect to arrive at efficient, more general computational methods for sensitivity analysis and parameter tuning for probabilistic networks. Our complexity results in fact suggest that further research should concentrate on tuning a limited number of parameters, in networks where inference is tractable.

The paper is organized as follows. After briefly reviewing the basic concepts involved in sensitivity analysis and parameter tuning in Section 2, we present some preliminaries from complexity theory in Section 3 and formally define several variants of the tuning problem in Section 4. We give a general completeness proof for these problems in Section 5. We further address some special, restricted cases of these problems in Section 6. The paper ends with our concluding observations

in Section 7.

2 Sensitivity analysis and tuning

A probabilistic network $\mathbf{B} = (\mathbf{G}, \Gamma)$ includes a directed acyclic graph $\mathbf{G} = (\mathbf{V}, \mathbf{A})$, where $\mathbf{V} = \{V_1, \ldots, V_n\}$ models a set of stochastic variables and $\mathbf{A}$ models the (in)dependences between them, and a set of parameter probabilities Γ, capturing the strengths of the relationships between the variables. The network models a joint probability distribution $\Pr(\mathbf{V}) = \prod_{i=1}^{n} \Pr(v_i \mid \pi(V_i))$ over its variables, where $\pi(V)$ denotes the parents of V in $\mathbf{G}$. We will use $\Pr(C = c \,|\, \mathbf{E} = \mathbf{e})$ to denote the probability of the value c of the output variable C, given an instantiation $\mathbf{e}$ to the set of evidence variables $\mathbf{E}$, which will be abbreviated as $\Pr(c \mid \mathbf{e})$. We will denote a particular set of parameter probabilities as $\mathbf{X} \subseteq \Gamma$, and we will use X to denote a single parameter. We will use $\mathbf{x}$ and x to denote the *combination of values* of a set of parameters, respectively the value of a single parameter. In sensitivity analysis and parameter tuning, we are interested in the effect of changes in the parameter probabilities $\mathbf{X}$ on an output probability for a designated variable C. The *sensitivity function* $f_{\Pr(c|\mathbf{e})}(\mathbf{X})$ expresses the probability of the output in terms of the parameter set $\mathbf{X}$. We will omit the subscript if no ambiguity can occur.

In a one-way sensitivity analysis, we measure the sensitivity of an output probability of interest with respect to a single parameter. The parameter under consideration is systematically varied from 0 to 1 and the other parameters from the same CPT are co-varied such that their mutual proportional relationship is kept constant [20]. Thus, if the parameter $X = \Pr(b_i \,|\, \rho)$ (denoting the conditional probability of the value b_i of the variable B given a particular configuration ρ of B's parents) is varied from 0 to 1, the other parameters $Pr(b_j \,|\, \rho)$ for the variable B are varied such that

$$\Pr(b_j \,|\, \rho)(X) = \Pr(b_j \,|\, \rho) \cdot \frac{1 - X}{1 - \Pr(b_i \,|\, \rho)}$$

for any value b_j other than b_i. Under the condition of covariation, the sensitivity function $f(X)$ is a quotient of two linear functions [7] and takes the form

$$f(X) = \frac{c_1 \cdot X + c_2}{c_3 \cdot X + c_4}$$

where the constants can be calculated from the other parameter probabilities in the network.

A one-way sensitivity analysis can be extended to measure the effect of the simultaneous variation of *two* parameters on the output [5]. The sensitivity function then generalizes to

$$f(X_1, X_2) = \frac{c_1 \cdot X_1 \cdot X_2 + c_2 \cdot X_1 + c_3 \cdot X_2 + c_4}{c_5 \cdot X_1 \cdot X_2 + c_6 \cdot X_1 + c_7 \cdot X_2 + c_8}$$

In this function, the terms $c_1 \cdot X_1 \cdot X_2$ and $c_5 \cdot X_1 \cdot X_2$ capture the interaction effect of the parameters on the output variable. This can further be generalized to *n-way* sensitivity analyses [6, 2] where multiple parameters are varied simultaneously. While higher-order analyses can reveal synergistic effects of variation, the results are often difficult to interpret [20].

For performing a one-way sensitivity analysis, efficient algorithms are available that build upon the observation that for establishing the sensitivity of an output probability it suffices to determine the constants in the associated sensitivity function. The simplest method for this purpose is to compute, from the network, the probability of interest for up to three values for the parameter under study; using the functional form of the function to be established, a system of linear equations is obtained, which is subsequently solved [7]. For the network computations involved, any standard propagation algorithm can be used. A more efficient method determines the required constants by propagating information through a junction tree, similar to the standard junction-tree propagation algorithm [12]. This method requires a very small number of inward and outward propagations in the tree to determine either the constants of all sensitivity functions that relate the probability of interest to any one of the network parameters, or to determine the sensitivity functions for any output probability in terms of a single parameter. Both algorithms are exponential in the size of the network, yet have a polynomial running time for networks of bounded treewidth.

Closely related to *analyzing* the effect of variation of parameters on the output—and often the next step after performing such an analysis—is *tuning* the parameters, such that the output has the desired properties. The output may need to satisfy particular constraints, e.g. $\Pr(c \mid \mathbf{e}) \geq q$, $\Pr(c_1 \mid \mathbf{e}) / \Pr(c_2 \mid \mathbf{e}) \geq q$ or $\Pr(c_1 \mid \mathbf{e}) - \Pr(c_2 \mid \mathbf{e}) \geq q$, for a particular value q. There are a number of algorithms to determine the solution space for a set of parameters given such constraints [2]. The computational complexity of these algorithms is always exponential in the treewidth w of the graph (i.e., the size of the largest clique in the jointree), yet varies from $O(c^w)$ for single parameter tuning, to $O(n \cdot \prod_{i=1}^{k} F(X_i) \cdot c^w)$ for tuning n parameters, where c is a constant, k is the number of CPTs that include at least one of the parameters being varied, and $F(X_i)$ denotes the size of the i-th CPT. Note that the tuning problem is related to the inference problem in so-called *credal networks* [8], where each variable is associated with sets of probability measures, rather than single values as in Bayesian networks. This problem has been proven $\mathsf{NP}^{\mathsf{PP}}$-complete [9].

Often, we want to select a combination of values for the

parameters that satisfies the constraints on the output probability of interest, but has minimal impact on the other probabilities computed from the network. In other cases, we want the modification to be as small as possible. In other words, we want to find a tuning that not merely satisfies the constraints, but is also *optimal*, either with respect to the minimal amount of parameter change needed, or the minimal change in the joint probability distribution induced by the parameter change. Here we discuss two typical distance measures between joint probability distributions, namely those proposed by Kullback and Leibler [13], and Chan and Darwiche [3].

The distance measure introduced by Chan and Darwiche [3], denoted by D_{CD}, between two joint probability distributions $\Pr_{\mathbf{x}}$ and $\Pr_{\mathbf{x}'}$ is defined as:

$$D_{CD}(\Pr_{\mathbf{x}}, \Pr_{\mathbf{x}'}) \stackrel{def}{=} \ln \max_{\omega} \frac{\Pr_{\mathbf{x}}(\omega)}{\Pr_{\mathbf{x}'}(\omega)} - \ln \min_{\omega} \frac{\Pr_{\mathbf{x}}(\omega)}{\Pr_{\mathbf{x}'}(\omega)}$$

where ω is taken to range over the joint probabilities of the variables in the network. The Kullback-Leibler measure [13], denoted by D_{KL}, is defined as:

$$D_{KL}(\Pr_{\mathbf{x}}, \Pr_{\mathbf{x}'}) \stackrel{def}{=} \sum_{\omega} \Pr_{\mathbf{x}}(\omega) \ln \frac{\Pr_{\mathbf{x}}(\omega)}{\Pr_{\mathbf{x}'}(\omega)}$$

Calculating either distance between two distributions is intractable in general. It can be proven that calculating D_{CD} is NP-complete and that calculating D_{KL} is PP-complete[1]. The *Euclidean distance* is a convenient way to measure the *amount of change* needed in $\mathbf{x}$ to go from $\Pr_{\mathbf{x}}$ to $\Pr_{\mathbf{x}'}$. This distance, denoted by D_E, is defined as:

$$D_E(\mathbf{x}, \mathbf{x}') \stackrel{def}{=} \sqrt{\sum_{x_i \in \mathbf{x}, x_i' \in \mathbf{x}'} (x_i - x_i')^2}$$

The Euclidean distance depends only on the parameters that are changed and can be calculated in $O(|\mathbf{X}|)$.

3 Complexity theory

In the remainder, we assume that the reader is familiar with basic concepts of computational complexity theory, such as the classes P and NP, and completeness proofs. For a thorough introduction to these subjects we refer to textbooks like [10] and [16]. In addition to these basic concepts, we use the complexity class PP (Probabilistic Polynomial time). This class contains languages L accepted in polynomial time by a *Probabilistic Turing Machine*. Such a machine augments the more traditional non-deterministic Turing Machine with a probability distribution associated with each state transition, e.g. by providing the machine with a tape, randomly filled with symbols [11]. If all choice points are binary and the probability of each transition is $\frac{1}{2}$, then the *majority* of the computation paths accept a string s if and only if $s \in L$.

A typical problem in PP (in fact PP-complete) is the INFERENCE problem [15, 18]: given a network $\mathbf{B}$, a variable V_1 in $\mathbf{V}$, and a rational number $0 \leq q \leq 1$, determine whether $\Pr(V_1 = v_1) \geq q$. Recall that $\Pr(V_1, \ldots, V_n) = \prod_{i=1}^{n} \Pr(V_i \mid \pi(V_i))$. To determine whether $\Pr(v_1) \geq q$, we sum over all marginal probabilities $\Pr(V_1, \ldots, V_n)$ that are consistent with v_1. This can be done using a Probabilistic Turing Machine in polynomial time. The machine calculates the multiplication of conditional probabilities $\Pr(V_i \mid \pi(V_i))$, $i = 1, \ldots, n$, choosing a computation path in which each variable V_i is assigned a value according to the conditional probability $\Pr(V_i \mid \pi(V_i))$. Each computation path corresponds to a specific joint value assignment, and the probability of arriving in a particular state corresponds with the probability of that assignment. At the end of this computation path, we accept with probability $\frac{1}{2} + (\frac{1}{q} - 1)\epsilon$, if the joint value assignment to $V_1, \ldots, V_n$ is consistent with v_1, and we accept with probability $(\frac{1}{2} - \epsilon)$ if the joint value assignment is *not* consistent with v_1. The majority of the computation paths (i.e., $\frac{1}{2} + \epsilon$) then arrives in an accepting state if and only if $\Pr(v_1) \geq q$.

Another concept from complexity theory that we will use in this paper is *oracle access*. A Turing Machine $\mathcal{M}$ has oracle access to languages in the class A, denoted as $\mathcal{M}^{\mathrm{A}}$, if it can query the oracle in one state transition, i.e., in $O(1)$. We can regard the oracle as a 'black box' that can answer membership queries in constant time. For example, $\mathsf{NP}^{\mathsf{PP}}$ is defined as the class of languages which are decidable in polynomial time on a non-deterministic Turing Machine with access to an oracle deciding problems in PP. Informally, computational problems related to probabilistic networks that are in $\mathsf{NP}^{\mathsf{PP}}$ typically combine some sort of *selecting* with *probabilistic inference*.

Not all real numbers are exactly computable in finite time. Since using real numbers may obscure the true complexity of the problems under consideration, we assume that all parameter probabilities in our network are rational numbers, thus ensuring that all calculated probabilities are rational numbers as well. This is a realistic assumption, since the probabilities are normally either assessed by domain experts or estimated by a learning algorithm from data instances. For similar reasons, we assume that $\ln(x)$ is approximated within a finite precision, polynomial in the binary representation of x.

[1] These results are not yet published but will be substantiated in a forthcoming paper.

4 Problem definitions

In the previous sections, we have encountered a number of computational problems related to sensitivity analysis and parameter tuning. To prove hardness results, we will first define decision problems related to these questions. Because of the formulation in terms of decision problems, all problems are in fact tuning problems.

PARAMETER TUNING
Instance: Let $\mathbf{B} = (\mathbf{G}, \Gamma)$ be a Bayesian network where Γ is composed of rational probabilities, and let Pr be its joint probability distribution. Let $\mathbf{X} \subseteq \Gamma$ be a set of parameters in the network, let C denote the output variable, and c a particular value of C. Furthermore, let $\mathbf{E}$ denote a set of evidence variables with joint value assignment $\mathbf{e}$, and let $0 \leq q \leq 1$.
Question: Is there a combination of values $\mathbf{x}$ for the parameters in $\mathbf{X}$ such that $\Pr_{\mathbf{x}}(c \mid \mathbf{e}) \geq q$?

PARAMETER TUNING RANGE
Instance: As in PARAMETER TUNING.
Question: Are there combinations of values $\mathbf{x}$ and $\mathbf{x}'$ for the parameters in $\mathbf{X}$ such that $\Pr_{\mathbf{x}}(c \mid \mathbf{e}) - \Pr_{\mathbf{x}'}(c \mid \mathbf{e}) \geq q$?

EVIDENCE PARAMETER TUNING RANGE
Instance: As in PARAMETER TUNING; furthermore let $\mathbf{e_1}$ and $\mathbf{e_2}$ denote two particular joint value assignments to the set of evidence variables $\mathbf{E}$.
Question: Is there a combination of values $\mathbf{x}$ for the parameters in $\mathbf{X}$ such that $\Pr_{\mathbf{x}}(c \mid \mathbf{e_1}) - \Pr_{\mathbf{x}}(c \mid \mathbf{e_2}) \geq q$?

MINIMAL PARAMETER TUNING RANGE
Instance: As in PARAMETER TUNING; furthermore let $r \in \mathbb{Q}^+$.
Question: Are there combinations of values $\mathbf{x}$ and $\mathbf{x}'$ for the parameters in $\mathbf{X}$ such that $D_E(\mathbf{x}, \mathbf{x}') \leq r$ and such that $\Pr_{\mathbf{x}}(c \mid \mathbf{e}) - \Pr_{\mathbf{x}'}(c \mid \mathbf{e}) \geq q$?

MINIMAL CHANGE PARAMETER TUNING RANGE
Instance: As in PARAMETER TUNING; furthermore let $s \in \mathbb{Q}^+$, and let D denote a distance measure for two joint probability distributions as reviewed in Section 2 .
Question: Are there combinations of values $\mathbf{x}$ and $\mathbf{x}'$ for the parameters in $\mathbf{X}$ such that $D(\mathbf{x}, \mathbf{x}') \leq s$ and $\Pr_{\mathbf{x}}(c \mid \mathbf{e}) - \Pr_{\mathbf{x}'}(c \mid \mathbf{e}) \geq q$?

MODE TUNING
Instance: As in PARAMETER TUNING; furthermore let $\top(\Pr(C))$ denote the mode of $\Pr(C)$.
Question: Are there combinations of values $\mathbf{x}$ and $\mathbf{x}'$ for the parameters in $\mathbf{X}$ such that $\top(\Pr_{\mathbf{x}}(C \mid \mathbf{e})) \neq \top(\Pr_{\mathbf{x}'}(C \mid \mathbf{e}))$?

Furthermore, we define EVIDENCE MODE TUNING, MINIMAL PARAMETER MODE TUNING, and MINIMAL CHANGE MODE TUNING corresponding to the PARAMETER TUNING variants of these problems.

5 Completeness results

We will construct a hardness proof for the PARAMETER TUNING RANGE problem. Hardness of the other problems can be derived with minimal changes to the proof construction. More specifically, we prove $\mathsf{NP}^{\mathsf{PP}}$-hardness of the PARAMETER TUNING RANGE-problem by a reduction from E-MAJSAT; this latter problem has been proven complete by Wagner [21] for the class $\mathsf{NP}^{\mathsf{PP}}$. We will use a reduction technique, similar to the technique used by Park and Darwiche [17] to prove $\mathsf{NP}^{\mathsf{PP}}$-hardness of the PARTIAL MAP-problem.

We first observe that all tuning problems from Section 4 are in $\mathsf{NP}^{\mathsf{PP}}$: given $\mathbf{x}$, $\mathbf{x}'$, q, r and s, we can verify all claims in polynomial time using a PP oracle, since inference is PP-complete [18]. For example, with the use of the oracle, we can verify in polynomial time whether $\Pr_{\mathbf{x}}(c \mid \mathbf{e}) - \Pr_{\mathbf{x}'}(c \mid \mathbf{e}) \geq q$, for a given $\mathbf{x}$, $\mathbf{x}'$, and q. Likewise, we can calculate the Euclidean distance of $\mathbf{x}$ and $\mathbf{x}'$ in polynomial time and verify that it is less than r. Determining whether a distance between two joint probability distributions is smaller than s is NP-complete (for the distance D_{CD} defined by Chan and Darwiche [3]) or PP-complete (for the distance D_{KL} defined by Kullback and Leibler [13]). Thus, we can non-deterministically compute an assignment to $\mathbf{X}$ and check (using a PP oracle) that the distance is smaller than s. Therefore, all problems are in $\mathsf{NP}^{\mathsf{PP}}$.

To prove hardness, we will reduce PARAMETER TUNING RANGE from E-MAJSAT, defined as follows:

E-MAJSAT
Instance: Let ϕ be a Boolean formula with n variables V_i ($1 \leq \ 1 \leq \ n$), grouped into two disjoint sets $\mathbf{V_E} = V_1, \ldots, V_k$ and $\mathbf{V_M} = V_{k+1}, \ldots, V_n$.
Question: Is there an instantiation to $\mathbf{V_E}$ such that for at least half of the instantiations to $\mathbf{V_M}$, ϕ is satisfied?

We construct a probabilistic network $\mathbf{B}_\phi$ from a given Boolean formula ϕ with n variables V_i and instantiation templates $\mathbf{V_E}$ and $\mathbf{V_M}$. For all variables V_i, in the formula ϕ, we create a matching stochastic variable V_i in $\mathbf{V}$ for the network $\mathbf{B}_\phi$, with possible values *true* and *false* with uniform distribution. These variables are roots in the network $\mathbf{B}_\phi$. We denote $X_i = \Pr(V_i = \mathit{true})$ as the *parameter* of V_i.

For each logical operator in ϕ, we create an additional stochastic variable in the network, whose parents are

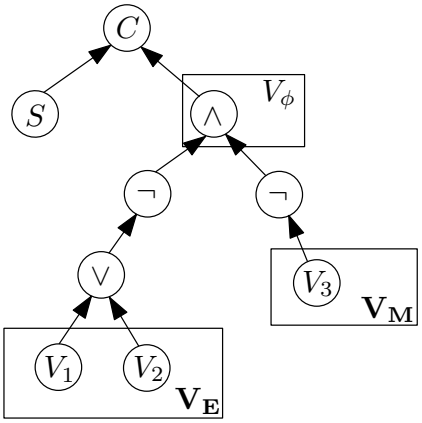


Figure 1: Example of construction

the corresponding sub-formulas (or single variable in case of a negation operator) and whose conditional probability table is equal to the truth table of that operator. For example, the $\wedge$-operator would have a conditional probability $\Pr(\wedge = \mathit{true}) = 1$ if and only if both its parents have the value *true*, and 0 otherwise. We denote the stochastical variable that is associated with the top-level operator in ϕ with V_ϕ. Furthermore, we add a variable S with values *true* and *false*, with uniform probability distribution where $X_S = \Pr(S = \mathit{true})$ is the parameter of S. Lastly, we have an output variable C, with parents S and V_ϕ and values *true* and *false*, whose CPT is equal to the truth table of the $\wedge$-operator. The set of parameters $\mathbf{X}$ in the Parameter Tuning Range problem now is defined to be $\{X_1, \ldots, X_k\} \cup X_S$, i.e., the parameters of the variables in $\mathbf{V_E}$ and the parameter of S. We set $q = \frac{1}{2}$.

Figure 1 shows the graphical structure of the probabilistic network constructed for the E-Majsat instance $(\phi, \mathbf{V_E}, \mathbf{V_M})$, where $\phi = \neg(V_1 \vee V_2) \wedge \neg V_3$, $\mathbf{V_E} = \{V_1, V_2\}$, and $\mathbf{V_M} = \{V_3\}$. Note that this E-Majsat instance is satisfiable with $V_1 = V_2 = F$; for that instantiation to $\mathbf{V_E}$, at least half of the possible instantiations to $\mathbf{V_M}$ will satisfy the formula.

Theorem 1. Parameter Tuning Range *is* $\mathsf{NP}^{\mathsf{PP}}$*-complete.*

Proof. Membership of $\mathsf{NP}^{\mathsf{PP}}$ can be proved as follows. Given $\mathbf{x}'$ and $\mathbf{x}$, we can verify whether $\Pr_{\mathbf{x}}(c \mid \mathbf{e}) - \Pr_{\mathbf{x}'}(c \mid \mathbf{e}) \geq q$ in polynomial time, given an oracle that decides Inference. Since Inference is PP-complete, this proves membership of $\mathsf{NP}^{\mathsf{PP}}$.

To prove hardness, we construct a transformation from the E-Majsat problem. Let $(\phi, \mathbf{V_E}, \mathbf{V_M})$ be an instance of E-Majsat, and let $\mathbf{B}_\phi$ be a probabilistic network, with parameters $\mathbf{x} = \{X_1, \ldots, X_k\} \cup X_S$, constructed as described above. Trivially, there exists a combination of parameter values $\mathbf{x}'$ such that $\Pr_{\mathbf{x}'}(C = \mathit{true}) = 0$, namely all assignments in which $X_S = 0$. In that case, at least one of the parents of C has the value *false* with probability 1 and thus $\Pr_{\mathbf{x}'}(C = \mathit{true}) = 0$.

On the other hand, if $\mathbf{x}$ includes $X_S = 1$, then $\Pr_{\mathbf{x}}(C = \mathit{true})$ depends on the values of $X_1, \ldots, X_k$. More in particular, there exist parameter values such that $\Pr_{\mathbf{x}}(C = \mathit{true}) \geq \frac{1}{2}$, if and only if $(\phi, \mathbf{V_E}, \mathbf{V_M})$ has a solution. We can construct a solution $\mathbf{x}$ by assigning 1 to X_S, 1 to all variables in $\{X_1, \ldots, X_k\}$ where the corresponding variable in $\mathbf{V_E}$ is set to *true*, and 0 where it is set to *false*. On the other hand, if $(\phi, \mathbf{V_E}, \mathbf{V_M})$ is *not* satisfiable, then $\Pr_{\mathbf{x}}(C = \mathit{true})$ will be less than $\frac{1}{2}$ for *any* parameter setting. Due to the nature of the CPTs of the 'operator' variables which mimic the truth tables of the operators, $\Pr_{\mathbf{x}}(C = \mathit{true}) = 1$ for a value assignment to the parameters that is consistent with a satisfying truth assignment to ϕ. If there does not exist a truth assignment to the variables in $\mathbf{V_E}$ such that the majority of the truth assignments to the variables in $\mathbf{V_M}$ satisfies ϕ, then there cannot be a value assignment to $\mathbf{X}$ such that $\Pr_{\mathbf{x}}(C = \mathit{true}) \geq \frac{1}{2}$. Thus, if we can decide whether there exist two sets of parameter settings $\mathbf{x}$ and $\mathbf{x}'$ such that in this network $\mathbf{B}_\phi$, $\Pr_{\mathbf{x}}(C = \mathit{true}) - \Pr_{\mathbf{x}'}(C = \mathit{true}) \geq q$, then we can answer $(\phi, \mathbf{V_E}, \mathbf{V_M})$ as well. This reduces E-Majsat to Tuning Parameter Range. □

Note that the constructed proof shows, that the Parameter Tuning Range problem remains $\mathsf{NP}^{\mathsf{PP}}$-complete, even if we restrict the set of parameters to constitute only prior probabilities, if all variables are binary, if all nodes have indegree at most 2, if the output is a singleton variable, and if there is no evidence. We will now show completeness proofs of the other problems.

Corollary 2. *All tuning problems defined in Section 4 are* $\mathsf{NP}^{\mathsf{PP}}$*-complete.*

Proof. We will show how the above construction can be adjusted to prove hardness for these problems.

- Parameter Tuning: From the above construct, leave out the nodes S and C, such that $\mathbf{x} = \{X_1, \ldots, X_k\}$. There is an instantiation $\mathbf{x}$ such that $\Pr_{\mathbf{x}}(V_\phi = \mathit{true}) \geq \frac{1}{2}$, if and only if $(\phi, \mathbf{V_E}, \mathbf{V_M})$ has a solution.
- Evidence Parameter Tuning Range: From the above construct, replace S with a singleton evidence variable E with values *true* and *false* and uniform distribution; denote $E = \mathit{true}$ as e_1 and $E = \mathit{false}$ as e_2 and let $\mathbf{x} = \{X_1, \ldots, X_k\}$.

$\Pr_{\mathbf{x}}(C = \mathit{true} \mid E = e_2) = 0$ for all possible parameter settings of $\mathbf{x}$. On the other hand, $\Pr_{\mathbf{x}}(V_\phi = \mathit{true}) \geq \frac{1}{2}$ and thus $\Pr_{\mathbf{x}}(C = \mathit{true} \mid E = e_1) \geq \frac{1}{2}$ if and only if $(\phi, \mathbf{V_E}, \mathbf{V_M})$ has a solution.

- Minimal Parameter Tuning Range and Minimal Change Parameter Tuning Range: These problems have Tuning Parameter Range as a special case (set $r, s = \infty$) and thus hardness follows by restriction.

- Mode Tuning: Since C has two values, $\Pr(C = \mathit{false}) = 1 - \Pr(C = \mathit{true})$. In particular, $\top(C) = \mathit{true}$ if $\Pr(C = \mathit{true}) \geq \frac{1}{2}$, and $\top(C) = \mathit{false}$ if $\Pr(C = \mathit{false}) \geq \frac{1}{2}$. If $X_S = 0$ then $\top(C) = \mathit{false}$. $\Pr(C = \mathit{true}) \geq \frac{1}{2}$, if and only if $\top(C) = \mathit{true}$.

Evidence Mode Tuning, Minimal Parameter Mode Tuning, and Minimal Change Mode Tuning: Apply similar construct modifications as with the corresponding Parameter Tuning problems. □

6 Restricted problem variants

In the previous section, we have shown that in the general case, Parameter Tuning Range is $\mathsf{NP}^{\mathsf{PP}}$-complete. In this section, the complexity of the problem is studied for restricted classes of instances. More in particular, we will discuss tuning problems in networks with bounded topologies and tuning problems with a bounded number of CPTs containing parameters to be tuned.

6.1 Bounded topologies

In this section we will show that restrictions on the topology of the network alone will not suffice to make the problem tractable. In fact, Parameter Tuning Range remains hard, even if $\mathbf{B}$ is a polytree. Similar results can be derived for the other problems. To prove NP-completeness of Parameter Tuning Range on polytrees, we reduce Maxsat to Parameter Tuning Range on polytrees, using a slightly modified proof from [17]. The (unweighted) Maxsat problem is defined as follows:

Maxsat
Instance: Let ϕ be a Boolean formula in CNF format, let $\mathbf{C}_\phi = C_1 \dots C_m$ denote its clauses and $\mathbf{V}_\phi = V_1 \dots V_n$ its variables, and let $1 \leq k \leq m$.
Question: Is there an assignment to the variables in ϕ, such that at least k clauses are satisfied?

We will construct a polytree network $\mathbf{B}$ as follows. For each variable in the formula, we create a variable in the network with values *true* and *false*, with uniform probability distribution. We denote the parameter of V_i as X_i as in the previous construct. We define a *clause selector variable* S_0 with values $c_1, \dots, c_m$ and uniform probability, i.e. $\Pr(S_0 = c_i) = \frac{1}{m}$. Furthermore, we define *clause satisfaction variables* S_i, with values $c_0, \dots, c_m$, associated with each variable. Every variable S_i has V_i and S_{i-1} as parents. Lastly, we define a variable V_S, with values *true* and *false*, with uniform probability distribution, and parameter X_S, and a variable C with values *true* and *false*, parents X_S and S_n. See Figure 2 for the topology of this network. The CPT for $S_i (i \geq 1)$ and C is given in Table 1. In this table, $T(V_i, j)$ and $F(V_i, j)$ are Boolean predicates that evaluate to 1 if the truth assignment to V_i satisfies, respectively does not satisfy, the j-th clause.

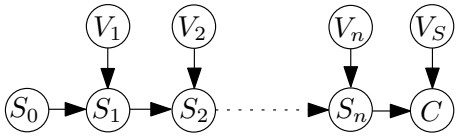


Figure 2: Construction with polytrees

$\Pr(S_i \mid V_i, S_{i-1})$		
S_{i-1}	$S_i = c_0$	$S_i \neq c_0$
c_0	1	0
c_j	$T(V_i, j)$	$F(V_i, j)$

$\Pr(C = T \mid V_S, S_n)$		
S_n	x_S	$\neg x_S$
c_0	1	0
c_j	1	1

Table 1: CPT for $\Pr(S_i \mid V_i, S_{i-1})$ and $\Pr(C \mid S_n, V_S)$

Theorem 3. Parameter Tuning Range *remains* NP*-complete if* $\mathbf{B}$ *is restricted to polytrees.*

Proof. Membership of NP is immediate, since we can decide Inference in polynomial time on polytrees. Given $\mathbf{x}'$ and $\mathbf{x}$, we can thus verify whether $\Pr_{\mathbf{x}'}(C = c) - \Pr_{\mathbf{x}}(C = c) \geq q$ in polynomial time.

To prove NP-hardness, we reduce Maxsat to Parameter Tuning Range. Let (ϕ, k) be an instance of Maxsat. From the clauses $\mathbf{C}_\phi$ and variables $\mathbf{V}_\phi$, we construct $\mathbf{B}_\phi$ as discussed above. Similarly as in the previous proof, if $X_S = 0$ then $\Pr(C = \mathit{true}) = 0$ for any instantiation to the parameters X_1 to X_n. If $X_S = 1$ then we observe the following. For every instantiation c_j to S_0, the probability distribution of S_i is as follows. $\Pr(S_i = c_0 \mid V_i) = 1$ if the instantiation to $V_1 \dots V_i$ satisfies clause c_j, and 0 otherwise. $\Pr(S_i = c_j \mid V_i) = 1$ if this instantiation does *not* satisfy clause c_j.

$$\Pr(S_i \mid x_i) = \begin{cases} S_i = c_0,\ V_{1\ldots i} \text{ satisfies } c_j & 1 \\ S_i = c_j,\ V_{1\ldots i} \text{ satisfies } c_j & 0 \\ S_i = c_0,\ V_{1\ldots i} \text{ does not satisfy } c_j & 0 \\ S_i = c_j,\ V_{1\ldots i} \text{ does not satisfy } c_j & 1 \\ \text{otherwise} & 0 \end{cases}$$

Of course, $\Pr(S_i)$ is conditioned on V_i and thus depends on X_i. For $X_i = 0$ or $X_i = 1$, either $\Pr(S_i = c_0) = 1$ or $\Pr(S_i = c_j) = 1$, for intermediate values of X_i the probability mass is shared between $\Pr(S_i = c_0)$ and $\Pr(S_i = c_j)$. But then $\Pr(S_n = c_0)$ is 1 for a particular clause selection c_j in S_0, if and only if the parameter setting to X_1 to X_n satisfies that clause. Due to the conditional probability table of C and $X_S = 1$, $\Pr_{\mathbf{x}}(C = \mathit{true}) = 1$ if and only if the parameter setting $\mathbf{x}$ satisfies that clause. Summing over S_0 yields $\Pr_{\mathbf{x}}(C = \mathit{true}) = \frac{k}{n}$, where k is the number of clauses that is satisfied by $\mathbf{x}$. Thus, a Parameter Tuning Range query with values 0 and $\frac{k}{n}$ would solve the Maxsat problem. This proves NP-hardness of Parameter Tuning Range on polytrees. □

6.2 Bounded number of CPTs

In the previous section we have shown that a restriction on the topology of the network in itself does not suffice to make parameter tuning tractable. In this section we will show that bounding the number of CPTs containing parameters in $\mathbf{X}$ in itself is not sufficient either. Note that trivial solutions to the Parameter Tuning may exist for particular subsets $\mathbf{X}$ of the set of parameter probabilities Γ. For example, if $\mathbf{X}$ constitutes all conditional probabilities $\Pr(C = c \,|\, \pi(C))$, for all configurations of parents of C, then a trivial solution would set all these parameters to q. If the number of parameters in $\mathbf{X}$ is logarithmic in the total number of parameter probabilities, i.e., $\mid \mathbf{X} \mid \leq p(\log \mid \Gamma \mid)$ for any polynomial p, then the problem is in P^{PP}, since we can try all combinations of parameter settings to 0 or 1 in polynomial time, using a PP-oracle.

If both the number of CPTs containing one or more parameters in the set $\mathbf{X}$ is bounded by a factor k (independent of the number of total number of parameter probabilities), and the indegree of the corresponding nodes is bounded, then Parameter Tuning is PP-complete. Hardness follows immediately since Parameter Tuning has Inference as a trivial special case (for zero parameters). We will prove membership of PP for this problem for a single parameter in a root node and show that the result also holds for a k-bounded number of CPTs with m parents. Similar observations can be made for the other tuning problems defined in Section 4.

Theorem 4. Parameter Tuning *is* PP*-complete if the number of CPTs containing parameters and the indegree of the corresponding nodes are bounded.*

Proof. First let us assume $k = 1$, i.e., all n parameters are taken from the CPT of a single node V. Furthermore, let us assume for now that V is a root node. To solve Parameter Tuning, we need to decide whether $\Pr(C = c) \geq q$ for a particular combination of values of the parameters in $\mathbf{X}$. Conditioning on V gives us $\sum_i \Pr(C = c \mid V = v_i) \cdot \Pr(V = v_i)$. Since $\sum_i \Pr(V = v_i) = 1$, $\Pr(C = c)$ is maximal for $\Pr(V = v_i) = 1$ for a particular v_i. Thus, if we want to decide whether $\Pr(C = c) \geq q$ for a particular combination of values of the parameters, then it suffices to determine whether this is the case when we set $\Pr(V = v_i) = 1$ for a particular parameter v_i.[2]

Using this observation, we will construct a Probabilistic Turing Machine $\mathcal{M}$ by combining several machines accepting Inference instances. At its first branching step, $\mathcal{M}$ either accepts with probability $\frac{1}{2}$, or runs, with probability $\frac{1}{2n}$, one of n Probabilistic Turing Machines $\mathcal{M}_i (i = 1, \ldots, n)$, which on input $\mathbf{B}_{\phi,i}$ (with $\Pr(V = v_i) = 1$) and q accept if and only if $\Pr(C = c) = q$. If any $\mathcal{M}_i$ accepts, then $\mathcal{M}$ accepts. The majority of computation paths of $\mathcal{M}$ accepts if and only if the Parameter Tuning instance is satisfiable. If V is *not* a root node, then we must branch over each parent configuration. For k CPTs with at most n parameters in each CPT and m incoming arcs, we need to construct a combined Probabilistic Turing Machine consisting of $O(n^{m^k})$ Probabilistic Turing Machines accepting instances of Inference. For bounded m and k, this is a polynomial number of machines and thus computation takes polynomial time. Thus, Parameter Tuning is in PP for a bounded number of CPTs containing parameters and a bounded indegree of the corresponding nodes m and k. □

7 Conclusion

In this paper, we have addressed the computational complexity of several variants of parameter tuning. Existing algorithms for sensitivity analysis and parameter tuning (see e.g. [2]) have a running time, exponential in both the treewidth of the graph and in the number of parameters varied. We have shown that parameter tuning is indeed hard, even if the network has a restricted polytree and if the number of parameters is bounded. We conclude, that Parameter Tuning is tractable only if *both* constraints are met, i.e., if probabilistic inference is easy *and* the number of parameters involved is bounded.

[2] If the number of parameters subject to tuning does not constitute all parameter probabilities in the CPT, then we need to test whether $\Pr(C = c) \geq q$ when all parameters have the value 0 as well.

Acknowledgments

This research has been (partly) supported by the Netherlands Organisation for Scientific Research (NWO). The authors wish to thank Hans Bodlaender and Gerard Tel for their insightful comments on earlier drafts of this paper. We wish to thank the anonymous reviewers for their thoughtful remarks.

References

[1] E. Castillo, J. M. Gutiérrez, and A. S. Hadi. Sensitivity analysis in discrete Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 27:412–423, 1997.

[2] H. Chan and A. Darwiche. Sensitivity analysis in Bayesian networks: From single to multiple parameters. In *Twentieth Conference on Uncertainty in Artificial Intelligence*, pages 67–75. AUAI Press, 2004.

[3] H. Chan and A. Darwiche. A distance measure for bounding probabilistic belief change. *International Journal of Approximate Reasoning*, 38(2):149–174, 2005.

[4] Th. Charitos and L. C. van der Gaag. Sensitivity analysis for threshold decision making with DBNs. In *22nd Conference on Uncertainty in Artificial Intelligence*, pages 72–79. AUAI Press, 2006.

[5] V. Coupé, L. C. van der Gaag, and J. D. F. Habbema. Sensitivity analysis: an aid for belief-network quantification. *Knowledge Engineering Review*, 15:1–18, 2000.

[6] V. M. H. Coupé, F. V. Jensen, U. B. Kjærulff, and L. C. van der Gaag. A computational architecture for n-way sensitivity analysis of Bayesian networks. Technical Report 102156, Aalborg University, 2000.

[7] V.M.H. Coupé and L.C. van der Gaag. Properties of sensitivity analysis of Bayesian belief networks. *Annals of Mathematics and Artificial Intelligence*, 36(4):323–356, 2002.

[8] Fabio G. Cozman. Credal networks. *Artificial Intelligence*, 120(2):199–233, 2000.

[9] C. P. de Campos and F. G. Cozman. The inferential complexity of bayesian and credal networks. In *International Joint Conference on Artificial Intelligence, Edinburgh, UK, 2005*, pages 1313–1318, 2005.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, 1979.

[11] J. T. Gill. Computational complexity of Probabilistic Turing Machines. *SIAM Journal of Computing*, 6(4), 1977.

[12] U. Kjærulff and L. C. van der Gaag. Making sensitivity analysis computationally efficient. In *Sixteenth Conference in Uncertainty in Artificial Intelligence*, pages 317–325. San Francisco: Morgan Kaufmann Publishers, 2002.

[13] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22:79–86, 1951.

[14] K. B. Laskey. Sensitivity analysis for probability assessments in Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 25:901–909, 1995.

[15] M. L. Littman, S. M. Majercik, and T. Pitassi. Stochastic boolean satisfiability. *Journal of Automated Reasoning*, 27(3):251–296, 2001.

[16] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[17] J. D. Park and A. Darwiche. Complexity results and approximation settings for MAP explanations. *Journal of Artificial Intelligence Research*, 21:101–133, 2004.

[18] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

[19] L. C. van der Gaag and S. Renooij. Analysing sensitivity data. In *Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 530–527. Morgan Kaufmann Publishers, San Francisco, 2001.

[20] L. C. van der Gaag, S. Renooij, and V. M. H. Coupé. Sensitivity analysis of probabilistic networks. In P. Lucas, J.A. Gámez, and A. Salmeron, editors, *Advances in Probabilistic Graphical Models, Studies in Fuzziness and Soft Computing*, volume 213, pages 103–124. Berlin: Springer Verlag, 2007.

[21] K. W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.

Small Sample Inference for Generalization Error in Classification Using the CUD Bound

Eric B. Laber and Susan A. Murphy
Department of Statistics
University of Michigan
Ann Arbor, MI 48104
{ laber, samurphy }@umich.edu

Abstract

Confidence measures for the generalization error are crucial when small training samples are used to construct classifiers. A common approach is to estimate the generalization error by resampling and then assume the resampled estimator follows a known distribution to form a confidence set [Kohavi 1995, Martin 1996,Yang 2006]. Alternatively, one might bootstrap the resampled estimator of the generalization error to form a confidence set. Unfortunately, these methods do not reliably provide sets of the desired confidence. The poor performance appears to be due to the lack of smoothness of the generalization error as a function of the learned classifier. This results in a non-normal distribution of the estimated generalization error. We construct a confidence set for the generalization error by use of a smooth upper bound on the deviation between the resampled estimate and generalization error. The confidence set is formed by bootstrapping this upper bound. In cases in which the approximation class for the classifier can be represented as a parametric additive model, we provide a computationally efficient algorithm. This method exhibits superior performance across a series of test and simulated data sets.

1 Introduction

Measures of uncertainty are particularly critical in the small sample setting where training set to training set variation is likely to be high. One example occurs in medical diagnostics in mental health in which one may want to classify patients as "low risk" or "high risk" for relapse based on observed medical history. In these settings the signal to noise ratio is likely to be small. Thus, both the estimated generalization error and the confidence in this estimate must be considered as the confidence might be very low due to the noise in the data. In addition, in this setting data must be collected from clinical trials and is subsequently expensive to obtain. The cost of these trials, coupled with notoriously low adherence rates, leads to data sets which are too small to admit a test set. Any inference for the generalization error must be derived from a single training set.

In this paper, we introduce a new method for obtaining confidence bounds on the generalization error in the small sample setting. The paper is organized as follows. Section 2 of the paper provides the framework for this research, and describes why this problem is difficult. We also survey several standard approaches to this problem and motivate the need for a new method. In section 3 we derive a new bound called the constrained uniform deviation bound (CUD bound hereafter) on the deviation between the estimated and true generalization errors and describe how to use this bound to form confidence sets. Section 4 gives an efficient algorithm for computing the new upper bound when the classifier can be represented as a parametric additive model. Section 5 describes an empirical study of several methods for constructing confidence sets and analyzes the results.

2 Framework

Our objective is to derive a confidence set for the generalization error under minimal assumptions. To this end, we only assume a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1,\ldots,n}$ of n iid draws from (unknown) distribution $\mathcal{F}$ over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is any feature space and $\mathcal{Y} = \{-1, 1\}$ is the label space. We also assume that there is a space of potential classifiers $\mathcal{G}$ from which we choose our classifier f using some criterion that depends on our training set $\mathcal{D}$. Equally important is what we *do not* assume. We make no assumptions

about the structure of $\mathcal{G}$, nor do we assume that the Bayes classifier belongs to $\mathcal{G}$. Finally, we do not assume the existence of an underlying "true" deterministic function $y = g(x)$ which generates the data.

Given a training set $\mathcal{D}$ we choose a classifier $f \in \mathcal{G}$ using some criterion (discussed in detail below). The classifier f aims to predict the value of a label y given feature x. Thus, a natural measure of the performance of f is its *generalization error* $\xi(f) = \mathbb{E}_{(X,Y)\sim\mathcal{F}}\mathcal{I}\{f(X) \neq Y\}$. That is, the probability that f will fail to correctly predict label Y from feature X.

When data are abundant, inference for the generalization error is straightforward. In this setting, one partitions the available data into two sets $\mathcal{L}$ and $\mathcal{T}$ called a learning set and a testing set respectively. Estimation is done by choosing classifier $f \in \mathcal{G}$ using learning set $\mathcal{L}$ and then using empirical estimate $\hat{\xi}_{test}(f) = \frac{1}{\#\mathcal{T}} \sum_{(x_i,y_i)\in\mathcal{F}} \mathcal{I}\{f(x_i) \neq y_i\}$. The exact distribution of $\hat{\xi}_{test}$ is known and confidence sets can be computed numerically [Langford 2005(1)]. Moreover, $\hat{\xi}_{test}(f)$ is, under mild regularity conditions, asymptotically normal, making standard asymptotic inference applicable.

The convenient properties of $\hat{\xi}_{test}(f)$ are a direct consequence of the independent test set $\mathcal{T}$. We define the small sample setting as one in which the data are too meager to admit a test set. Without a test set one is forced to use the same data both to train and evaluate the classifier. Estimates and confidence sets for the generalization error must be constructed using resampling (e.g. bootstrap, cross-validation, etc.). However, when the training sets are small, the training set to training set distribution of these estimates may be highly non-normal, thus complicating the construction of confidence sets. This is particularly the case here as the generalization error $\xi(f)$ is a non-smooth function of f.

It is worth pointing out that there exist alternative motivations for seeking confidence sets in the absence of a testing set. The formation of a test set essentially "wastes" observations which are not used in the training of the classifier. The omission of these observations can significantly deteriorate the performance of the learned classifier. Another motivation is the existence of learning algorithms which assume that training accuracy correctly reflects generalization error when selecting a classifier. Using high quality bounds on the generalization error in the algorithm can boost performance [Langford 2005(2)] particularly when the training data is small.

Relevant Work

Much of the work on small sample inference for the generalization error has focused on point estimates via resampling. Popular point estimates include the .632+ estimator given by Efron and Tibshirani [Efron 1995], the $LV1O^*$ estimate given by Weiss [1991], and more recently the bolstered error estimate of Braga-Neto and Dougherty [Dougherty 2004]. A nice survey of generalization error estimates is given by Schiavo and Hand [Schiavo 2000].

In the literature, a variety of general approaches have been suggested for constructing confidence sets for the generalization error in the absence of a test set. We discuss them in turn.

1. *Known distribution.* One approach to this problem is to assume the resampled estimator follows a known distribution Φ [Kohavi 1995, Yang 2006]. Typically Φ is taken to be normal, so that a $1-\delta$ confidence set for estimated generalization error $\hat{\xi}(f)$ of classifier f is given by

$$\hat{\xi}(f) \quad \pm \quad \Phi^{-1}\left(\frac{\delta}{2}\right)\sqrt{\nu}/\sqrt{n^*},$$

   where ν is some estimate of the standard error and n^* is the effective sample size[1] used to construct the resampled estimator. This approach mimics the case where $\hat{\xi}(f)$ is constructed from a training set and then a normal approximation would have been justified for the estimator of the generalization error based on an independent test set.

2. *Bayesian Approach.* This approach mimics the case where $\hat{\xi}(f)$ is constructed from a training set and then using an independent test set, a Bayesian approach is used to construct a posterior confidence set. Here we specify a Beta prior π on the generalization error $\xi(f)$ [Martin 1995].

$$\xi(f) \quad \sim \quad \beta(\gamma, \lambda).$$

   If an independent test set $\mathcal{T} = \{(x_i, y_i)\}_{i=1,\ldots,m}$ were available then random variables $z_i = \mathcal{I}(f(x_i) = y_i)$ would be iid $bern(\xi)$. The posterior $P(\xi(f)|z_1, \ldots, z_m)$ is also a Beta distribution,

$$\xi(f)|z_1, \ldots, z_m \quad \sim \quad \beta(\gamma + m\hat{\xi}(f), \lambda + m - m\hat{\xi}(f))$$

   Note that the posterior depends only on the number of misclassified points, $m\hat{\xi}(f)$ on the test set.

[1] Here we use effective sample size to mean the expected number of unique points used in estimation during a single step of a resampling procedure.

Suppose we do not have a test set. Then given resampled estimate $\hat{\xi}$, we can view $(n-n^*)\hat{\xi}(f)$ as the misclassified examples where n and n^* are the full and effective sample sizes. One can think of this as training on a set of size n^* and testing on a set of size $n-n^*$. Using this idea, the resampled model is given by,

$$\begin{aligned} \xi(f) &\sim \beta(\gamma,\lambda) \\ \xi(f)|z_1,\ldots,z_m &\sim \beta(\nu,\eta) \\ \nu &= \gamma+\hat{\xi}(f)(n-n^*) \\ \eta &= \lambda+(1-\hat{\xi}(f))(n-n^*). \end{aligned}$$

The posterior given above can be used to construct confidence intervals for $\xi(f)$.

3. *Direct Estimation.* In this approach we repeatedly resample the data and from each resampled dataset we construct a point estimate of the generalization error (this may require further resampling). This process generates a sequence of estimates for the generalization error. A confidence set can be constructed, for example, by taking the quantiles of the generated sequence of resampled estimators.

The methods described above can work well in some settings. Distribution-based methods tend to work well when distributional assumptions are met, providing tight confidence sets and accurate coverage. However, these methods can be non-robust to large deviations from distributional assumptions and can suffer from unacceptable type I errors. Direct estimation avoids distributional assumptions but is unstable for small samples because of non-smooth 0-1 loss. These problems are illustrated in the experiments sections of the paper.

Ideally confidence sets for the generalization error should satisfy the following criteria.

1. The confidence set should deliver the promised coverage for a wide variety of examples.
2. The confidence set should be efficiently computable.
3. The confidence set should be theoretically justified.

Given multiple algorithms for constructing confidence sets satisfying the above, algorithms that produce confidence sets of smallest size are preferred. We propose a new approach and compare this approach to the above methods using the first and second criteria in the experiments section.

3 The CUD Bound

We construct an upper bound on the deviation between the training error and generalization error of a classifier using ideas from the large literature on constructing bounds on the so-called *excess-risk* of a selected classifier [Shawe-Taylor 1998, Barlett and Mendelson 2006]. Also we allow the use of a convex surrogate as in Zhang [2004] for the 0-1 loss. In particular we use the idea of constructing a bound for the supremum of the deviation between the training error and generalization error over a small subset of the approximation space $\mathcal{G}$. This small subset is determined by the convex surrogate loss, $\mathcal{L}:(\mathcal{D},f)\to\mathbb{R}$ used to construct the classifier ($\hat{f}=\arg\min_{f\in\mathcal{G}}\mathcal{L}(\mathcal{D},f)$). A variety of surrogate loss functions may be used, a common loss function is squared error loss, given by $\mathcal{L}(\mathcal{D},f)=\frac{1}{n}\sum_{(x_i,y_i)\in\mathcal{D}}(y_i-f(x_i))^2$. Some other examples are the hinge loss, logistic loss, exponential loss and penalized quadratic loss.

The intuition for the upper bound is as follows. Suppose we knew (we don't) the limiting value of $\hat{f}$, say f^*, as the amount of training data becomes infinite. f^* may not be the Bayes classifier. Also notice that f^* *does not* depend on the training set. In addition, since we can think of $\hat{f}$ as being concentrated near f^*, this motivates the use of f^* as the anchor for the small subset over which we take the supremum. In particular, we take the supremum over the set of classifiers belonging to a neighborhood of f^*. The size of this neighborhood should depend on how far $\hat{f}$ is from f^* in terms of the difference in loss, $\mathcal{L}(\mathcal{D},\hat{f})-\mathcal{L}(\mathcal{D},f^*)$.

To fix notation, let $\hat{\xi}_{\mathcal{S}}(f)$ be the empirical error of f on data set $\mathcal{S}$, that is,

$$\hat{\xi}_{\mathcal{S}}(f)=\frac{1}{\#\mathcal{S}}\sum_{(x_i,y_i)\in\mathcal{S}}\mathcal{I}\{f(x_i)\neq y_i\}$$

so that $\hat{\xi}_{\mathcal{D}}(\hat{f})$ is the *training error*. We have the following result.

CUD Bound. *If α_n is any positive function of the size n of the training set $\mathcal{D}$ and $g(x)=(x+1)\wedge 1$ then:*

$$\left|\hat{\xi}_{\mathcal{D}}(\hat{f})-\xi(\hat{f})\right|\leq\sup_{f\in\mathcal{G}}\left|\hat{\xi}_{\mathcal{D}}(f)-\xi(f)\right|g(\alpha_n(\mathcal{L}(\mathcal{D},f^*)-\mathcal{L}(\mathcal{D},f)))$$

Proof. First notice that $g(x)\equiv 1\;\forall\, x\in\mathbb{R}_+$. Then, since $\hat{f}=\arg\min_{f\in\mathcal{G}}\mathcal{L}(\mathcal{D},f)$ we must have

$$\mathcal{L}(\mathcal{D},f^*)-\mathcal{L}(\mathcal{D},\hat{f})\geq 0$$

so that,

$$g(\alpha_n(\mathcal{L}(\mathcal{D},f^*)-\mathcal{L}(\mathcal{D},\hat{f})))=1.$$

The result follows from noticing,

$$\begin{aligned}\big|\hat{\xi}_{\mathcal{D}}(\hat{f}) - \xi(\hat{f})\big| &= \big|\hat{\xi}_{\mathcal{D}}(\hat{f}) - \xi(\hat{f})\big| \times \\ &\qquad g(\alpha_n(\mathcal{L}(\mathcal{D}, f^*) - \mathcal{L}(\mathcal{D}, \hat{f}))) \\ &\leq \sup_{f\in\mathcal{G}} \big|\hat{\xi}_{\mathcal{D}}(f) - \xi(f)\big| \times \\ &\qquad g(\alpha_n\big(\mathcal{L}(\mathcal{D}, f^*) - \mathcal{L}(\mathcal{D}, f))),\end{aligned}$$

with the last inequality following because $\hat{f} \in \mathcal{G}$. □

The intuition here is that we consider a small neighborhood around f^* with α_n chosen so that the size of the neighborhood shrinks at the same rate $\mathcal{L}(\mathcal{D}, \hat{f}) \to \mathcal{L}(\mathcal{D}, f^*)$ as the size of $\mathcal{D}$ tends to infinity. It is the function g which restricts the domain over which the supremum is taken. In particular,

$$\begin{aligned}&|\hat{\xi}_{\mathcal{D}}(\hat{f}) - \xi(\hat{f})| \\ &\leq \sup_{f\in\mathcal{G}} \big|\hat{\xi}_{\mathcal{D}}(f) - \xi(f)\big| g(\alpha_n\big(\mathcal{L}(\mathcal{D}, f^*) - \mathcal{L}(\mathcal{D}, f))) \\ &\leq \sup_{f\in\mathcal{G}} \big|\hat{\xi}_{\mathcal{D}}(f) - \xi(f)\big| \mathcal{I}\big(\mathcal{L}(\mathcal{D}, f^*) \geq \mathcal{L}(\mathcal{D}, f) - \alpha_n^{-1}\big) \\ &= \sup_{f\in\mathcal{W}} \big|\hat{\xi}_{\mathcal{D}}(f) - \xi(f)\big|,\end{aligned}$$

where,

$$\mathcal{W} = \big\{f \in \mathcal{G} : \mathcal{L}(\mathcal{D}, f^*) \geq \mathcal{L}(\mathcal{D}, f) - \alpha_n^{-1}\big\}.$$

From the preceding set of inequalities we see that g serves as a continuous approximation to the indicator function. The rate, α_n, will depend on the complexity of $\mathcal{G}$ and the training set size, n. For example if $\mathcal{G}$ has low complexity such as a smoothly parameterized class with a finite dimensional parameter then the rate can be expected to be on the order of n. We now present the main result which follows immediately from the upper bound.

Proposition 1. *Let $\delta \in (0, 1]$, then if $\mathcal{Q}_{1-\delta}$ is the $1-\delta$ quantile of,*

$$\sup_{f\in\mathcal{G}} \big|\hat{\xi}_{\mathcal{D}}(f) - \xi(f)\big| g(\alpha_n\big(\mathcal{L}(\mathcal{D}, f^*) - \mathcal{L}(\mathcal{D}, f))),$$

then $P\big(\big|\hat{\xi}_{\mathcal{D}}(\hat{f}) - \xi(\hat{f})\big| \leq Q_{1-\delta}) \geq 1 - \delta$.

Thus, the problem is reduced to computing $\mathcal{Q}_{1-\delta}$. This is done using the bootstrap.

3.1 Computing $\mathcal{Q}_{1-\delta}$

For any fixed δ we estimate $\mathcal{Q}_{1-\delta}$ with its bootstrap estimate $\hat{\mathcal{Q}}_{1-\delta}$. A bootstrap sample $\mathcal{D}^{(b)}$ from $\mathcal{D}$ is an iid draw of size n from the distribution which puts equal mass on each point in $\mathcal{D}$. The bootstrap estimate is constructed by treating the original sample $\mathcal{D}$ as the true population and each bootstrap draw as a random draw of size n from the true population. The mapping between population and bootstrap quantities is straightforward and given in the table below. The reader is referred to [Efron 1994] or [Hall 1992] for details.

Population	Bootstrap
$\xi(f)$	$\hat{\xi}_{\mathcal{D}}(f)$
f^*	$\hat{f}$
$\mathcal{L}(\mathcal{D},)$	$\mathcal{L}(\mathcal{D}^{(b)},)$
$\hat{\xi}_{\mathcal{D}}(f)$	$\hat{\xi}_{\mathcal{D}^{(b)}}(f)$

Using the above, the procedure to compute $\hat{\mathcal{Q}}_{1-\delta}$ is as follows.

1. Construct B bootstrap datasets $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(B)}$ from $\mathcal{D}$.

2. For $b = 1, \ldots, B$ compute:

$$\begin{aligned}\mathcal{Q}^{(b)} &= sup_{f\in\mathcal{G}} \big|\hat{\xi}_{\mathcal{D}^{(b)}}(f) - \hat{\xi}_{\mathcal{D}}(f)\big| \times \\ &\qquad g(\alpha_n(\mathcal{L}(\mathcal{D}^{(b)}, \hat{f}) - \mathcal{L}(\mathcal{D}^{(b)}, f))).\end{aligned}$$

3. Set $\hat{\mathcal{Q}}_{1-\delta}$ to the $1-\delta$ sample quantile of $\mathcal{Q}^{(1)}, \ldots, \mathcal{Q}^{(B)}$.

We now substitute $\hat{\mathcal{Q}}_{1-\delta}$ into Proposition 1 in place of $\mathcal{Q}_{1-\delta}$ to construct the confidence set.

Initially the computation of the sup in step 2 above appears to be computationally intensive. Depending on the form of classifier space $\mathcal{G}$ and loss function $\mathcal{L}(,)$ taking this sup may prove to be computationally infeasible. However, when the loss function $\mathcal{L}(,)$ is convex and the classifier is approximated by a parametric additive model, computation is straightforward by way of a branch and bound algorithm which we discuss next.

4 An Efficient Algorithm For Parametric Additive Models

In this section we consider the space of classifiers

$$\mathcal{G} = \{f(x) = sign\bigg(\sum_{i=1}^{p} \beta_i b(\gamma_i, x)\bigg), \beta_i, \gamma_i \in \mathbb{R}\ \forall i\},$$

where $b(,)$ are basis functions indexed by the γ_i's and the β_i's are the basis coefficients. We also assume a convex surrogate loss function $\mathcal{L}(\mathcal{D}, f)$ where convexity refers to convexity in $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$ but not necessarily in $\boldsymbol{\gamma} = \gamma_1, \ldots, \gamma_p$. We assume that given training

set $\mathcal{D}$ we choose classifier

$$\begin{aligned}\hat{f}(x) = f(x;\hat{\boldsymbol{\beta}},\hat{\boldsymbol{\gamma}}) &= \arg\min_{\boldsymbol{\beta},\boldsymbol{\gamma}} \mathcal{L}(\mathcal{D}, f(;\boldsymbol{\beta},\boldsymbol{\gamma})) \\ &= \arg\min_{\boldsymbol{\beta}} \left(\arg\min_{\boldsymbol{\gamma}} \mathcal{L}(\mathcal{D}, f(;\boldsymbol{\beta},\boldsymbol{\gamma})) \right) \\ &= \arg\min_{\boldsymbol{\beta}} \mathcal{L}(\mathcal{D}, f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}}))\end{aligned}$$

We now hold $\hat{\boldsymbol{\gamma}}$ fixed and compute the upper bound as a supremum over $\boldsymbol{\beta}$.

CUD Bound (Parametric Additive Models). *If α_n is any positive function of the size n of the training set $\mathcal{D}$ and $g(x) = (x+1) \wedge 1$ then:*

$$\left|\hat{\xi}_{\mathcal{D}}(f(;\hat{\boldsymbol{\beta}},\hat{\boldsymbol{\gamma}})) - \xi(f(;\hat{\boldsymbol{\beta}},\hat{\boldsymbol{\gamma}}))\right|$$

is bounded above by:

$$\begin{aligned}&\sup_{\boldsymbol{\beta}\in\mathbb{R}^p} \left|\hat{\xi}_{\mathcal{D}}(f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}})) - \xi(f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}}))\right| \\ \times\quad & g(\alpha_n\big(\mathcal{L}(\mathcal{D}, f(;\boldsymbol{\beta}^*,\boldsymbol{\gamma}^*)) - \mathcal{L}(\mathcal{D}, f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}}))))\end{aligned}$$

The supremum is difficult to calculate because of the term with the absolute values; it is the absolute difference between two sums of indicator functions and is hence both non-smooth and non-convex. We now discuss how to transform this problem into a series of convex optimization problems with linear constraints.

To begin, suppose $\mathcal{D}^{(b)}$ is a bootstrap dataset drawn from $\mathcal{D}$. For any training point (x_i, y_i) in $\mathcal{D}$, let ϕ_i be the number of copies of (x_i, y_i) in $\mathcal{D}^{(b)}$. Then $0 \leq \phi_i \leq n$ for all i and the distribution of each ϕ_i is binomial with size n and probability $\frac{1}{n}$. Then we can write,

$$\begin{aligned}&n\left|\hat{\xi}_{\mathcal{D}^{(b)}}(f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}})) - \hat{\xi}_{\mathcal{D}}(f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}}))\right| \\ =\ & \left|\sum_{(x_i,y_i)\in\mathcal{D}} (\phi_i - 1)\mathcal{I}\{f(x_i;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}}) \neq y_i\}\right| \\ =\ & \left|\sum_{(x_i,y_i)\in\mathcal{D}^{(b)}-\mathcal{D}} (\phi_i - 1)\mathcal{I}\{f(x_i;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}}) \neq y_i\}\right|.\end{aligned}$$

The above term does not depend on points with $\phi_i = 1$, that is, those points which were selected exactly once in the bootstrap resampling. We let $\mathcal{D}^{(b)} - \mathcal{D}$ denote the set of training points (x_i, y_i) that satisfy $\phi_i \neq 1$.

Notice that the number of points in $\mathcal{D}^{(b)} - \mathcal{D}$ is necessarily smaller than n which translates into computational savings. To see this, we will need to make use of the following equivalence relation.

Given $\boldsymbol{\beta}^1, \boldsymbol{\beta}^2 \in \mathbb{R}^p$ and $\mathcal{S}$ a subset of $\mathcal{D}$, we say $\boldsymbol{\beta}^1$ is congruent to $\boldsymbol{\beta}^2$ modulo $\mathcal{S}$ if $f(x;\boldsymbol{\beta}^1,\hat{\gamma}) = f(x;\boldsymbol{\beta}^2,\hat{\gamma})$ for all x such that $(x, y) \in \mathcal{S}$ for some y (i.e. $\boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$ lead to the same classification on $\mathcal{S}$). Congruency modulo $\mathcal{S}$ defines an equivalency relation and hence partitions $\mathbb{R}^p$. For any fixed bootstrap dataset $\mathcal{D}^{(b)}$ let $\mathcal{S}$ be the collection of distinct points in $\mathcal{D}^{(b)} - \mathcal{D}$ and let $\mathcal{M}_1, \ldots, \mathcal{M}_R$ be the subsequent equivalence classes. Then, for any equivalence class $\mathcal{M}_i$,

$$\left|\hat{\xi}_{\mathcal{D}^{(b)}}(f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}})) - \hat{\xi}_{\mathcal{D}}(f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}}))\right| \equiv C(\mathcal{M}_i) \forall \boldsymbol{\beta} \in \mathcal{M}_i,$$

where $C(\mathcal{M}_i)$ is a constant. Then referring back to step 2 of the bootstrap procedure outlined in section 3 we can write,

$$\begin{aligned}&\mathcal{Q}^{(b)} = sup_{\boldsymbol{\beta}\in\mathbb{R}^p}\left|\hat{\xi}_{\mathcal{D}}(f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}}) - \hat{\xi}_{\mathcal{D}^{(b)}}(f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}}))\right| \\ &\qquad \times g(\alpha_n(\mathcal{L}(\mathcal{D}^{(b)}, f(;\hat{\boldsymbol{\beta}},\hat{\boldsymbol{\gamma}})) - \mathcal{L}(\mathcal{D}^{(b)}, f(;\hat{\boldsymbol{\beta}},\hat{\boldsymbol{\gamma}}))))) \\ &= sup_i C(\mathcal{M}_i) \\ &\times \sup_{\boldsymbol{\beta}\in\mathcal{M}_i} g(\alpha_n(\mathcal{L}(\mathcal{D}^{(b)}, f(;\hat{\boldsymbol{\beta}},\hat{\boldsymbol{\gamma}})) - \mathcal{L}(\mathcal{D}^{(b)}, f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}})))) \\ &= \sup_i C(\mathcal{M}_i) \\ &\times g(\alpha_n(\mathcal{L}(\mathcal{D}^{(b)}, f(;\hat{\boldsymbol{\beta}},\hat{\boldsymbol{\gamma}})) - \inf_{\boldsymbol{\beta}\in\mathcal{M}_i} \mathcal{L}(\mathcal{D}^{(b)}, f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}})))),\end{aligned}$$

with the last equality following from the monotonicity of g. Since $\mathcal{L}$ is convex in $\boldsymbol{\beta}$ this would be a series of convex optimization problems if membership in $\mathcal{M}_i$ could be expressed as a set of convex constraints. We now show how this can be done.

Let $\boldsymbol{\beta}_{\mathcal{M}_i}$ be a representative member of equivalence class $\mathcal{M}_i$. Then for any other $\boldsymbol{\beta} \in \mathcal{M}_i$ we must have,

$$f(x_j;\boldsymbol{\beta}_{\mathcal{M}_i},\hat{\boldsymbol{\gamma}}) \sum_{k=1}^{p} \beta_k b(\hat{\gamma}_k, x_j) \quad \geq \quad 0$$

for all x_j such that $(x_j, y) \in \mathcal{D}^{(b)} - \mathcal{D}$ for some y. Since $f(x_j;\boldsymbol{\beta}_{\mathcal{M}_i},\hat{\boldsymbol{\gamma}})$ does not depend on $\boldsymbol{\beta}$ we see that membership to $\mathcal{M}_i$ is equivalent to satisfying a series of linear constraints, which is clearly convex. We have shown that calculation of $\mathcal{Q}^{(b)}$ amounts to a series of R convex optimization problems.

In practice, even though n is small (e.g. $n \leq 50$) R can be very large (on the order of 2^n in the worst case) making direct computation of all R convex optimization problems infeasible. Fortunately, exhaustive computation *can* be done using a branch and bound algorithm.

To use branch and bound we must recursively partition the search space. To do this we arbitrarily label the feature vectors in $\mathcal{D}^{(b)} - \mathcal{D}$ by $x_1, \ldots, x_d$. The root of the tree represents all of $\mathbb{R}^p$. The first left child represents the set of all $\boldsymbol{\beta} \in \mathbb{R}^p$ so that $\sum_{k=1}^{p} \beta_k b(\hat{\gamma}_k, x_1) \geq 0$ while the first right child represents all $\boldsymbol{\beta} \in \mathbb{R}^p$ so that $\sum_{k=0}^{p} \beta_i b(\hat{\gamma}_k, x_1) < 0$. In general, if $\mathcal{S}$ is a region defined by a node at level j of the tree, then its left child represents the subspace of

$\mathcal{S}$ which satisfies $\boldsymbol{\beta} \in \mathcal{S}$ and $\sum_{k=0}^{p} \beta_k b(\hat{\gamma}_k, x_{j+1}) \geq 0$, similarly, its right child is the subspace of $\mathcal{S}$ which satisfies $\boldsymbol{\beta} \in \mathcal{S}$ and $\sum_{k=1}^{p} \beta_k b(\hat{\gamma}_k, x_{j+1}) < 0$. Notice that the terminal nodes of this tree are either infeasible or one of the equivalence classes $\mathcal{M}_1, \ldots, \mathcal{M}_R$.

To complete the branch and bound algorithm we need to define upper and lower bounds on the value of objective function,

$$\mathcal{O}(\boldsymbol{\beta}) = \left|\hat{\xi}_{\mathcal{D}^{(b)}}(f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}}) - \hat{\xi}_{\mathcal{D}}(f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}}))\right| \times g(\alpha_n(\mathcal{L}(\mathcal{D}^{(b)}, f(;\hat{\boldsymbol{\beta}},\hat{\boldsymbol{\gamma}})) - \mathcal{L}(\mathcal{D}^{(b)}, f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}})))).$$

In particular, given region $\mathcal{S}$ of $\mathbb{R}^p$ defined by a node on the tree representing our partition, we require an upper bound $\mathcal{U}(\mathcal{S})$ on $\mathcal{O}(\boldsymbol{\beta})$,

$$\sup_{\boldsymbol{\beta} \in \mathcal{S}} \mathcal{O}(\boldsymbol{\beta}) \leq \mathcal{U}(\mathcal{S}).$$

The upper bound $\mathcal{U}(\mathcal{S})$ is constructed by bounding the two terms in $\mathcal{O}(\boldsymbol{\beta})$ separately. An upper bound on $|\hat{\xi}_{\mathcal{D}^{(b)}}(f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}})) - \hat{\xi}_{\mathcal{D}}(f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}}))|$ can be obtained by noticing that if $\mathcal{S}$ is a region of $\mathbb{R}^p$ defined by a node at level j, then the classification of the first j points in $\mathcal{D}^{(b)} - \mathcal{D}$ is fixed on $\mathcal{S}$. The upper bound is constructed by assuming the worst performance possible on the remaining $d-j$ points. To bound the second term, we compute,

$$sup_{\boldsymbol{\beta} \in \mathcal{S}} g(\alpha_n(\mathcal{L}(\mathcal{D}^{(b)}, f(;\hat{\boldsymbol{\beta}})) - \mathcal{L}(\mathcal{D}^{(b)}, f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}})))),$$

which is a convex optimization problem. The upper bound $\mathcal{U}(\mathcal{S})$ is the product of these two upper bounds.

In addition, we require the following lower bound,

$$L(\mathcal{S}) \leq sup_{\boldsymbol{\beta} \in \mathcal{S}} \mathcal{O}(\boldsymbol{\beta}).$$

The lower bound $L(\mathcal{S})$ is obtained by plugging any feasible point in $\mathcal{S}$ into $\mathcal{O}(\boldsymbol{\beta})$. In practice, a natural choice is the $argsup$ of the second term in the objective function, which has already been computed during the construction of $\mathcal{U}(\boldsymbol{\beta})$.

This algorithm running on a standard desktop with a sample size of $n = 50$ and using a total of 500 bootstrap samples to construct a confidence set runs in a only a few minutes. While this algorithm is still an NP-hard problem,[2] in practice it is significantly less computationally intensive than evaluating all possible classifications. The reason is that the function g allows for significant reduction of the search space by restricting attention only to classifiers within a fixed distance of selected the classifier $\hat{f}$. To see this, notice that in the construction of $\mathcal{U}(\mathcal{S})$, if the term

$$sup_{\boldsymbol{\beta} \in \mathcal{S}} g(\alpha_n(\mathcal{L}(\mathcal{D}^{(b)}, f(;\hat{\boldsymbol{\beta}})) - \mathcal{L}(\mathcal{D}^{(b)}, f(;\boldsymbol{\beta},\hat{\boldsymbol{\gamma}})))),$$

is non-positive we can remove the region $\mathcal{S}$ from our search space since we know $\mathcal{O}(\hat{\beta}) \geq 0$.

5 Experiments

In this section we describe a set of experiments designed to test the coverage and diameter of confidence sets constructed using the CUD bound. To form a baseline for comparison we also construct confidence sets using a handful of methods found in the literature. These consist of two non-parametric bootstrap methods, a normal approximation using the bootstrap [Kohavi 1995], a normal approximation using CV [Yang 2006], a generalization of a Bayesian approach [Martin 1995], and the inverted binomial [Langford 2005].[3] An online supplemental appendix provides a full description of these methods, the simulation study, and provides links to the source code (www.stat.lsa.umich.edu/∼laber/UAI2008.html).
The confidence sets were constructed for sample sizes of $n = 30$ and $n = 50$ using the datasets given in table 1. All the data sets have binary labels and continuous

Dataset	Motivation
Spam	Not Simulated
Ionosphere	Not Simulated
Heart	Not Simulated
Diabetes	Not Simulated
Abalone	Not Simulated
Liver	Not Simulated
Mammogram	Not Simulated
Magic	Not Simulated
Donut (Simulated)	f^* close to Bayes
Outlier (Simulated)	f^* far from Bayes
χ^2_{Small} (Simulated)	$f^* = 0$, low noise, far from Bayes
Three Points (Simulated)	$\hat{\xi}(\hat{f})$ highly non-normal, f^* far from Bayes

Table 1: Datasets used in experiments along with the reason for their inclusion. Here f^* is the limiting value of the classifier as the amount of training data becomes infinite.

features. The real datasets can be found at the UCI data repository (www.ics.uci.edu/mlearn). The simulated data sets were designed to investigate the performance of the new procedure in several scenarios of interest.

[2] Update: In the case of linear classification with squared error loss, current work in progress has proved this runs in polynomial time, on the order of $\mathcal{O}(n^{VC(\mathcal{G})})$.

[3] In this approach we are plugging in a resampled estimate and the effective sample size. That is,we are acting as if we had a test set.

For this series of experiments we fit a linear classifier using squared error loss. That is,

$$\mathcal{G} = \{f(x) = sign(\sum_{i=1}^{p} \beta_i \phi_i(x)),\ \beta_i \in \mathbb{R}\}$$

where the ϕ_i are transformations of the features (in all cases ϕ_i was either a polynomial or projection onto a number of principle components). The surrogate loss function is given by

$$\mathcal{L}(\mathcal{D}, f(;\boldsymbol{\beta}) = \sum_{(x_i,y_i)\in\mathcal{D}} (\sum_{j=1}^{p} \beta_j \phi_j(x_i) - y_i)^2,$$

and the scaling factor α_n used in g is chosen to be n. For each data set the following procedure was used to estimate coverage probabilities.

1. Randomly divide data into training set $\mathcal{D}$ and testing set $\mathcal{T}$.
   - If the data set is real, randomly select n training points for $\mathcal{D}$ and use the remainder for $\mathcal{T}$.
   - If the data set is simulated, generate n data points for $\mathcal{D}$ and generate 5000 data points for $\mathcal{T}$.
2. Build classifier $\hat{f} = \arg\min_{f\in\mathcal{G}} \mathcal{L}(\mathcal{D}, f)$ using training data $\mathcal{D}$.
3. For each procedure use the training data $\mathcal{D}$ and chosen classifier $\hat{f}$ to construct confidence set with target coverage .950. The confidence interval is centered at $\hat{\xi}_{\mathcal{D}}(\hat{f})$ taken to be the .632 estimate [Efron 1983] in the the competing methods and the training error in the CUD bound.[4]
4. Using the $\xi(\hat{f}) = \hat{\xi}_{\mathcal{T}}(\hat{f})$ we record the coverage for each procedure.

The above steps are repeated 1000 times for each data set and the average coverage computed. The results are shown in tables 2 and 3 with the following abbreviations: quantile bootstrap (BS1) [Efron 1994], corrected boostrap (BS2) [Efron 1994], Yang's CV estimate (Y) [Yang 2006], Kohavi's normal approximation (K) [Kohavi 1996], a generalization of a Bayesian approach from Martin (M) [Martin 1996], and the inverted binomial of Langford (L) [Langford 2005].

The results for the $n = 50$ case are similar and the results are given in tables 4 and 5 of the appendix.

The results in tables 2 and 3 show promising results for the new method. It was the only method to provide the desired coverage on all ten datasets and, in addition, possessed the smallest diameter for two of them. The CUD bound confidence set is conservative, as must be expected by its construction. However, the diameter of the new confidence set is far from trivial, even for this extremely small sample size.

[4]The competing methods experience a significant reduction in performance if they are centered at the training error.

	n = 30						
Data	CUD	BS1	BS2	K	M	Y	L
Spam	1.00	.478	.983	.782	.632	.996	.636
Donut	.999	.880	.908	.631	.633	.937	.620
Ionosphere	.998	.605	.926	.816	.757	.994	.747
Heart	.999	.406	.981	.718	.475	.998	.458
Diabetes	1.00	.654	.900	.912	.986	.997	.984
Outlier	.988	.884	.736	.808	.838	.910	.961
3 Pt.	.993	.827	.717	.844	.896	.753	.952
Abalone	.983	.963	.737	.972	.991	.998	.997
Liv.	.961	.964	.878	.980	1.00	1.00	1.00
Magic	.998	.918	.920	.961	.982	.991	.992
χ^2	.985	.958	.786	.961	.982	.996	1.00
Mammogram	1.00	.678	.994	.646	.426	.983	.406

Table 2: Average coverage of new method and competitors. Target coverage level was .950, those meeting or exceeding this level are highlighted in orange.

	n = 30						
Data	CUD	BS1	BS2	K	M	Y	L
Spam	.469	.432	.432	.315	.314	.451	.354
Donut	.485	.590	.590	.324	.324	.414	.364
Ionosphere	.437	.429	.429	.301	.296	.501	.337
Heart	.459	.468	.468	.319	.319	.433	.359
Diabetes	.433	.305	.305	.307	.310	.312	.350
Outlier	.555	.491	.491	.328	.329	.456	.368
3 Pt.	.508	.476	.476	.318	.317	.457	.356
Abalone	.617	.314	.314	.331	.331	.504	.371
Liver	.559	.372	.372	.327	.326	.485	.366
Magic	.606	.307	.307	.300	.283	.464	.324
χ^2	.595	.330	.330	.329	.330	.501	.351
Mammogram	.464	.532	.532	.321	.321	.421	.361

Table 3: Average diameter of confidence set constructed using new method and competitors. Method with smallest diameter and having correct coverage is highlighted in yellow.

It is also of interest to learn when competing methods perform poorly. Figure 1 plots the empirical coverage of the top four methods against the mean absolute value of the training error's kurtosis (standardized central fourth moment subtract 3) for each of the nine data sets listed in table 1. The kurtosis for a normal distribution is 0, and we use absolute kurtosis as a measure of deviation from normality. Figure 1 shows a trend of decreasing coverage as deviation from normality increases for the three distribution based methods. This suggests these methods may not be robust

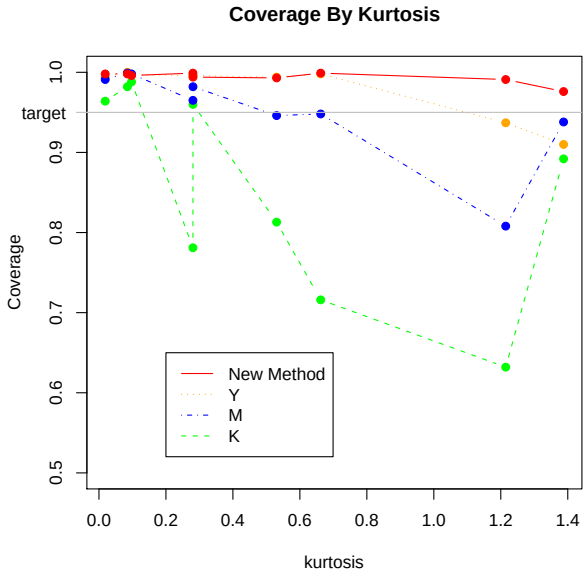


Figure 1: Shows coverage of top four methods by absolute value of kurtosis of the training error.

to non-normal training error. In particular we see that the method K (which has strongest normality assumptions) is most sensitive to deviation from normality. In contrast, the CUD bound method has stable, if conservative, coverage, regardless of the normality of training error.

6 Conclusions and Future Work

In this paper we introduced a new method for constructing confidence sets for the generalization error in the small sample setting in classification. This bound is derived under the minimal assumption of a training set of *iid* draws from some unknown distribution $\mathcal{F}$. In addition, we provided a computationally efficient algorithm for constructing this new confidence set when the classifier is approximated by a parametric additive model. In preliminary experiments, the new confidence set exhibited superior coverage while maintaining a small diameter on a catalog of test and simulated data sets. Moreover, it was demonstrated that confidence sets based on distributional assumptions may not be robust to deviation from normality in the training samples.

Much work remains to be done. First, we need to provide theoretical guarantees for the level of confidence. We conjecture that a correct choice of α_n will depend on the complexity of the approximation space used to construct the classifier. It is well known that bootstrap estimators perform better when estimating smooth functions. Selecting $\alpha_n = \infty$ is similar to standard quantile bootstrap (BS1) which is ineffective in the classification setting because it bootstraps the non-smooth 0-1 loss. This suggests an upper bound on α_n. Conversely, a smaller choice of α_n leads to a bootstrapped estimate of a smoother quantity but also introduces conservatism. For example, setting $\alpha_n = 0$ is equivalent to a uniform deviation bound on the training error which can often be trivially loose. This suggests a lower bound on α_n as well. Thus, α_n must strike a balance between smoothness required by the bootstrap and conservatism.

References

[1] P. Bartlett, M. Jordan, and J. McAuliffe. Convexity, classification, and risk bounds. *J. Amer. Statis. Assoc.*, 101(473):138–156, Mar 2006.

[2] P. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, 3:463–482, 2002.

[3] Ulisses Braga-Neto and Edward Dougherty. Bolstered error estimation. *Pattern Recognition*, 37:1267–1281, 2004.

[4] B. Efron. Estimating the error rate of a prediction rule: Improvement on cross-validation. *J. Amer. Statis. Assoc.*, 78(382):316–331, 1983.

[5] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, May 1994.

[6] B. Efron and R. Tibshirani. Cross-validation and the bootstrap: Estimating the error of a prediction rule. Technical report, 1995.

[7] Peter Hall. *The Bootstrap and Edgeworth Expansion*. Springer Series in Statistics. Springer, New York, 1992.

[8] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145, 1995.

[9] J. Langford. Combining train set and test set bounds. *ICML*, 2002.

[10] J. Langford. Tutorial on practical prediction theory for classification. *J. Machine Learning Research*, 6:273–306, 2005.

[11] J. Kent Martin and Daniel S. Hirschberg. Small sample statistics for classification error rates II: Confidence intervals and significance tests. Technical Report ICS-TR-96-22, 1996.

[12] Rosa A. Schiavo and David J. Hand. Ten more years of error rate research. *International Statistical Review*, 68(3):295–310, 2000.

[13] S.M. Weiss. Small sample error rate estimation for k-nn classifiers. *Transactions on Pattern Analysis and Machine Intelligence*, 13(3):285–289, Mar 1991.

[14] Yuhong Yang. Comparing learning methods for classification. *Statistica Sinica*, 16(2):635–657, 2006.

[15] Xi Zhang and Kang G. Shin. Statistical analysis of feedback-synchronization signaling delay for multicast flow control. In *INFOCOM*, pages 1152–1161, 2001.

Appendix: Experimental Results for $n = 50$

	n = 50						
Data	CUD	BS1	BS2	K	M	Y	L
Spam	1.00	.172	.987	.415	.632	.996	.636
Donut	.998	.837	.904	.450	.633	.882	.620
Ionosphere	1.00	.292	.975	.575	.757	.985	.747
Heart	.999	.072	.999	.186	.475	.998	.458
Diabetes	.999	.654	.900	.912	.986	.984	.984
Out.	.984	.805	.893	.702	.838	.933	.961
3 Pt.	.972	.800	.812	.648	.896	.624	.952
Abalone	.988	.960	.832	.962	.991	.999	.997
Liver	.986	.888	.888	.974	1.00	.988	1.00
Magic	1.00	.824	.823	.950	.982	.998	.992
χ^2	.999	.955	.791	.960	.982	.996	1.00
Mammogram	.999	.317	.999	.146	.426	.991	.406

Table 4: Average coverage of new method and competitors. Target coverage level was .950, those meeting or exceeding this level are highlighted in orange.

	n = 50						
Data	CUD	BS1	BS2	K	M	Y	L
Spam	.418	.377	.377	.250	.314	.329	.354
Donut	.448	.529	.529	.258	.324	.331	.364
Ionosphere	.386	.364	.364	.237	.296	.315	.337
Heart	.409	.436	.436	.254	.319	.443	.359
Diabetes	.433	.365	.365	.247	.310	.358	.350
Out.	.442	.439	.439	.262	.329	.328	.368
3 Pt.	.440	.413	.413	.253	.317	.357	.356
Abalone	.523	.248	.248	.264	.331	.391	.371
Liver	.499	.304	.304	.260	.326	.382	.366
Magic	.561	.209	.209	.227	.283	.350	.324
χ^2	.401	.265	.265	.236	.330	.389	.351
Mammogram	.419	.511	.511	.255	.321	.320	.361

Table 5: Average diameter of confidence set constructed using new method and competitors. Method with smallest diameter and having correct coverage is highlighted in yellow.

Discovering Cyclic Causal Models by Independent Components Analysis

Gustavo Lacerda
Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Peter Spirtes
Joseph Ramsey
Department of Philosophy
Carnegie Mellon University
Pittsburgh, PA 15213

Patrik O. Hoyer
Dept. of Computer Science
University of Helsinki
Helsinki, Finland

Abstract

We generalize Shimizu et al's (2006) ICA-based approach for discovering linear non-Gaussian acyclic (LiNGAM) Structural Equation Models (SEMs) from causally sufficient, continuous-valued observational data. By relaxing the assumption that the generating SEM's graph is acyclic, we solve the more general problem of linear non-Gaussian (LiNG) SEM discovery. LiNG discovery algorithms output the distribution equivalence class of SEMs which, in the large sample limit, represents the population distribution. We apply a LiNG discovery algorithm to simulated data. Finally, we give sufficient conditions under which only one of the SEMs in the output class is "stable".

1 Linear SEMs

Linear structural equation models (SEMs) are statistical causal models widely used in the natural and social sciences (including econometrics, political science, sociology, and biology) [1].

The variables in a linear SEM can be divided into two sets, the error terms (typically unobserved), and the substantive variables. For each substantive variable $\mathbf{x}_i$, there is a linear equation with $\mathbf{x}_i$ on the left-hand-side, and the direct causes of $\mathbf{x}_i$ plus the corresponding error term on the right-hand-side.

Each SEM with jointly independent error terms can be associated with a directed graph (abbreviated as DG) that represents the causal structure of the model and the form of the linear equations. The vertices of the graph are the substantive variables, and there is a directed edge from $\mathbf{x}_i$ to $\mathbf{x}_j$ just when the coefficient of $\mathbf{x}_i$ in the structural equation for $\mathbf{x}_j$ is non-zero. [1]

[1] Traditionally, SEMs with acyclic graphs are called "recursive", and SEMs with cyclic graphs "non-recursive" [15]. We avoid this usage, and use "acyclic" or "cyclic" instead.

1.1 The model, with an illustration

Let $\mathbf{x}$ be the random vector of substantive variables, $\mathbf{e}$ be the vector of error terms, and B be the matrix of linear coefficients for the substantive variables. Then the following equation describes the linear SEM model:

$$\mathbf{x} = B\mathbf{x} + \mathbf{e} \tag{1}$$

For example, consider the model defined by:

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{e}_1 \\ \mathbf{x}_2 &= 1.2\mathbf{x}_1 - 0.3\mathbf{x}_4 + \mathbf{e}_2 \\ \mathbf{x}_3 &= 2\mathbf{x}_2 + \mathbf{e}_3 \\ \mathbf{x}_4 &= -\mathbf{x}_3 + \mathbf{e}_4 \\ \mathbf{x}_5 &= 3\mathbf{x}_2 + \mathbf{e}_5 \end{aligned} \tag{2}$$

Note that the coefficient of each variable on the left-hand-side of the equation is 1.

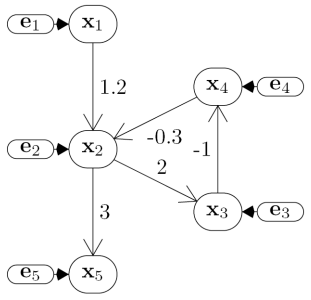


Fig. 1: Example 1

$\mathbf{x}$ can also be expressed directly as a linear combination of the error terms, as long as $I - B$ is invertible. Solving for $\mathbf{x}$ in Eq. 1 gives $\mathbf{x} = (I - B)^{-1}\mathbf{e}$. If we let $A = (I - B)^{-1}$, then $\mathbf{x} = A\mathbf{e}$. A is called the reduced form matrix (in the terminology of Independent

Components Analysis (see Section 3.1), it is called the "mixing matrix").

The distributions over the error terms in a SEM, together with the linear equations, entail a joint distribution over the substantive variables. This joint distribution can be interpreted in terms of physical processes, as shown next.

1.2 Interpretating linear SEMs

These equations (contained in matrix equation (1)) can be given several different interpretations. Under one class of interpretations, they are a set of equations satisfied by a set of variables $\mathbf{x}$ in equilibrium. With some further assumptions, the B matrix in the simultaneous equations (a.k.a. "equilibrium equations") also represents the coefficients in a set of dynamical equations describing a deterministic dynamical system.

Fisher [5] gave one such interpretation as follows: There is a relatively long observation period of length 1, and a much shorter "reaction" lag of length $\Delta\theta = 1/n$. The observed variable is the vector $\bar{\mathbf{x}}[t]$, defined as the average of $\mathbf{x}$ over the observation period starting at t:

$$\bar{\mathbf{x}}[t] \equiv \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}[t + k\Delta\theta] \tag{3}$$

Suppose that the underlying dynamical equations are:

$$\mathbf{x}[t + k\Delta\theta] = B_{dyn}\mathbf{x}[t + (k-1)\Delta\theta] + \mathbf{e} \tag{4}$$

where $\mathbf{e}$ is constant over the observation period (but may differ for different units in the population, e.g. different observation periods).

Fisher showed that, in the limit as $\Delta\theta$ approaches 0, there is a $B_{equil} = B_{dyn}$ such that:

$$\bar{\mathbf{x}}[t] = B_{equil}\bar{\mathbf{x}}[t] + \mathbf{e} \tag{5}$$

if and only if the modulus of each eigenvalue of B_{dyn} is less than or equal to 1, and no eigenvalue is equal to 1.

The assumptions underlying this model are fairly strong, but commonly made in econometrics, and defended by Fisher [5].

A simpler, but similar interpretation with similar consequences is the one in which the observed value $\mathbf{x}$ is the state in which the dynamical system converged, rather than its average over an observation period.

1.2.1 Dealing with self-loops

The LiNG discovery algorithms presented in this paper (described in section 4) output a set of directed graphs that do not contain any "self-loops" (edges from a vertex to itself) [2], i.e. the B matrices output by our LiNG discovery algorithms have all zeros in the diagonal. This is because it is impossible to determine the values of the diagonal entries of the B matrix from equilibrium data alone.

In the underlying dynamical equations, it may be that for some index a, $\mathbf{x}_a[t + (k-1)\Delta\theta]$ affects $\mathbf{x}_a[t + k\Delta\theta]$ (i.e. $b_{a,a} \neq 0$). Our goal is to recover the coefficients that both represent the distribution of $\bar{\mathbf{x}}$ and correctly predict the effects of manipulations. A manipulation of a variable $\mathbf{x}_i$ to a distribution P is modeled by replacing the dynamical equation for $\mathbf{x}_i$ by a new dynamical equation $\mathbf{x}_i[t + k\Delta\theta] = \mathbf{e'}_i$, where $\mathbf{e'}_i$ has distribution P [10].

For these purposes, the following argument sketches why the underdetermination of the diagonal of B_{equil} by the equilibrium data is not a problem, as long as $b_{a,a} \neq 1$ in the underlying dynamical equations.

The equation for $\bar{\mathbf{x}}_a$ has the form:

$$\bar{\mathbf{x}}_a = b_{a,a}\bar{\mathbf{x}}_a + \sum_{k\neq a, k=1}^{n} b_{a,k}\bar{\mathbf{x}}_k + e_a \tag{6}$$

If $b_{a,a} \neq 1$, it is possible to rewrite this as:

$$\bar{\mathbf{x}}_a - b_{a,a}\bar{\mathbf{x}}_a = \sum_{k\neq a, k=1}^{n} b_{a,k}\bar{\mathbf{x}}_k + e_a \tag{7}$$

$$\bar{\mathbf{x}}_a = \frac{1}{1-b_{a,a}}\left(\sum_{k\neq a, k=1}^{n} b_{a,k}\bar{\mathbf{x}}_k + e_a\right) = \sum_{k=1}^{n} b'_{a,k}\bar{\mathbf{x}}_k + e'_a \tag{8}$$

where $b'_{a,a} = 0$. The modified system of equations containing Equation 8 is represented by a graph that has no self-loops, and has a different underlying dynamical equation in which the coefficient for $\mathbf{x}_a[t + (k-1)\Delta\theta]$ in the equation for $\mathbf{x}_a[t + k\Delta\theta]$ is zero.

Note that in the second equation, the error term e_a has been rescaled by $1/(1-b_{a,a})$ to form a new error term e'_a and when $(I-B)^{-1}$ is taken to form the reduced form coefficients, the coefficients corresponding to e_a in the first set of equations will be multiplied by $(1 - b_{a,a})$, and the two changes cancel each other out.

Now, if we consider the original dynamical system and the one that results from setting the diagonal of B to zero (as above), it is sometimes the case that one dynamical system satisfies the conditions for the dynamical equations to approach the simultaneous equations

[2] Fisher argues that self-loops are not realistic, but these arguments are not entirely convincing.

in the limit, while the other one does not, because the magnitude of the coefficients in the equation for $\mathbf{x}_a[t]$ are different. If both forms satisfy Fisher's conditions, then the act of manipulating any variable to a fixed distribution for all t makes the two sets of dynamical equations have equivalent limiting simultaneous equations.

1.2.2 Self-loops with coefficient 1

Unfortunately, the case where $b_{a,a} = 1$ cannot be handled in the same way, since $1/(1 - b_{a,a})$ is infinite. If $b_{a,a} = 1$, then there may be no equivalent form without a self-loop (or more precisely, the corresponding equations without a self-loop may require setting the variance of some error terms to zero). The case where $b_{a,a} = 1$ is a genuine problem that we do not currently have a solution for. For the purposes of this paper, we assume that no self-loops have a coefficient of 1.

As Dash has pointed out [4], there are cases where the simultaneous equations have a different graph than the underlying dynamical equations, and hence the graph that represents the simultaneous equations cannot be used to predict the effects of a manipulation of the underlying dynamical system. In [4], Dash presents two such examples. In both of them, in effect, B_{dyn} has a 1 in the diagonal.

2 The problem and its history

2.1 The problem of DG causal discovery

Using the interpretations from 1.2, we can frame the problem as follows: given samples of the equilibrium distribution of a LiNG process whose observed variables form a causally sufficient set [3], find the set of SEMs that describe this distribution, under the assumption that it is non-empty.

2.2 Richardson's Cyclic Causal Discovery (CCD) Algorithm

While many algorithms have been suggested for discovering (equivalence classes of) directed acyclic graphs (DAGs) from data, for general linear directed graphs (DGs) only one provably correct algorithm was known (until now), namely Richardson's Cyclic Causal Discovery (CCD) algorithm.

CCD outputs a "partial ancestral graph" (PAG) that represents both a set of directed graphs that entail the same set of zero partial correlations for all values of the linear coefficients, and features common to those directed graphs (such as ancestor relations). The algorithm performs a series of statistical tests of zero partial correlations to construct the PAG. The set of zero partial correlations that is entailed by a linear SEM with uncorrelated errors depends only upon the linear coefficients, and not upon the distribution of the error terms. Under some assumptions[4], in the large sample limit, CCD outputs a PAG that represents the true graph.

There are a number of limitations to this algorithm. First, the set of DGs contained in a PAG can be large, and while they all entail the same zero partial correlations (viz., those judged to hold in the population), they need not entail the same joint distribution or even the same covariances. Hence in some cases, the set represented by the PAG will include cyclic graphs that do not fit the data well. Therefore, even assuming that the errors are all Gaussian, it is possible to reduce the size of the set of graphs output by CCD, although in practice this can be intractable. For details on the algorithm, see [11].

3 Shimizu et al's approach for discovering LiNGAM SEMs

The "LiNGAM algorithm"[12], which uses Independent Components Analysis (ICA), reliably discovers a unique correct LiNGAM SEM, under the following assumptions about the data: the structural equations of the generating process are linear and can be represented by an acyclic graph; the error terms have non-zero variance; the samples are independent and identically distributed; no more than one error term is Gaussian; and the error terms are jointly independent.[5]

3.1 Independent Components Analysis (ICA)

Independent components analysis [3, 8] is a statistical technique used for estimating the mixing matrix A in equations of the form $\mathbf{x} = A\mathbf{e}$ ($\mathbf{e}$ is often called "sources" and written $\mathbf{s}$), where $\mathbf{x}$ is observed and $\mathbf{e}$ and A are not.

ICA algorithms find the invertible linear transforma-

[3] A set V of variables is causally sufficient for a population if and only if in the population every common direct cause of any two or more variables in V is in V. (For subtleties regarding this definition, see [13]).

[4] The assumptions are: the samples are independent and identically distributed, no error term has zero variance, the statistical tests for zero partial correlations are consistent, linearity of the equations, the existence of a unique reduced form, faithfulness (i.e. there are no zero partial correlations in the population that are not entailed for all values of the free parameters of the true graph), and that the error terms are uncorrelated.

[5] The error terms are typically not jointly independent if the set of variables is not causally sufficient.

tion $W = A^{-1}$ of the data X that makes the error distributions corresponding to the implied samples E of $\mathbf{e}$ maximally non-Gaussian (and thus, maximally independent). The matrix A can be identified up to scaling and permutation as long as the observed distribution is a linear, invertible mixture of independent components, at most one of which is Gaussian [3]. There are computationally efficient algorithms for estimating A [8].

3.2 The LiNGAM discovery algorithm

If we run an ICA algorithm on data generated by a linear SEM, the matrix W_{ICA} obtained will be a row-scaled, row-permuted version of $I - B$, where B is the coefficient matrix of the true model (this is a consequence of the derivation in Section 1.1). We are now left with the problem of finding the proper permutation and scale for the W matrix so that it equals $I - B$.

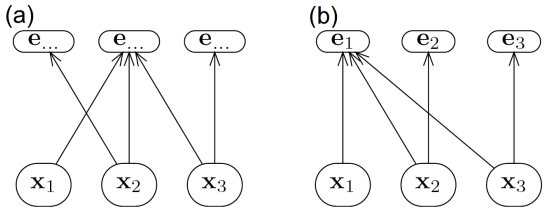


Fig. 2: After removing the edges whose coefficients are statistically indistinguishable from zero: (a) the raw W_{ICA} matrix output by ICA on a SEM whose graph is $\mathbf{x}_2 \rightarrow \mathbf{x}_1 \leftarrow \mathbf{x}_3$ (b) the corresponding $\widetilde{W}$ matrix, obtained by permuting the error terms in W_{ICA}

Since the order of the error terms given by ICA is arbitrary, the algorithm needs to correctly match each error term $\mathbf{e}_i$ to its respective substantive variable $\mathbf{x}_i$. This means finding the correct permutation of the rows of W_{ICA}. We know that the row-permutation of W_{ICA} corresponding to the correct model cannot have a zero in the diagonal (we call such permutations "inadmissible") because $W = I - B$, and the diagonal of B is zero.

Since, by assumption, the data was generated by a DAG, there is exactly one row-permutation of W_{ICA} that is admissible [12]. To visualize this, this constraint says that there is exactly one way to reorder the error terms so that every $\mathbf{e}_i$ is the target of a vertical arrow.[6]

In this example, swapping the first and second error terms is the only permutation that produces an admissible matrix, as seen in Fig. 2(b).

[6] Another consequence of acyclicity is that there will be no right-pointing arrows in this representation, provided that the $\mathbf{x}$s are topologically sorted w.r.t. the DAG.

After the algorithm finds the correct permutation, it finds the correct scaling, i.e. "normalizing" W by dividing each row by its diagonal element, so that the diagonal of the output matrix is all 1s (i.e. the coefficient of each error term is 1, as specified in Section 1).

Bringing it all together, the algorithm computes B by using $B = I - W'$, where $W' = normalize(\widetilde{W})$, $\widetilde{W} = RowPermute(W_{ICA})$ and $W_{ICA} = ICA(X)$.

Besides the fact that it determines the direction of every causal arrow, another advantage of LiNGAM over conditional-independence-based methods [13] is that the correctness of the algorithm does not require the faithfulness assumption.

For more details on the LiNGAM approach, see [12].

4 Discovering LiNG SEMs

The assumptions of the family of LiNG discovery algorithms described below (abbreviated as "LiNG-D") are the same as the LiNGAM assumptions, replacing the assumption that the SEM is acyclic with the weaker assumption that the diagonal of B_{dyn} contains no 1s. In this more general case, as in the acyclic case, candidate models are generated by finding all admissible matches of the error terms ($\mathbf{e}_i$'s) to the observed variables ($\mathbf{x}_i$'s). In other words, each candidate corresponds to a row-permutation of the W_{ICA} matrix that has a zeroless diagonal.

As in LiNGAM, the output is the set of admissible models. In LiNGAM, this set is guaranteed to contain a single model, thanks to the acyclicity assumption. If the true model has cycles, however, more than one model will be admissible.

The remainder of this section addresses the problem of finding the admissible models, given that ICA has finite data to work with.

4.1 Prune and solve Constrained n-Rooks

These algorithms generate candidate models by testing which entries of W_{ICA} are zero (i.e. pruning), and finding all admissible permutations based on that (i.e. solving Constrained n-Rooks, see Section 4.1.2). We call an algorithm "local" if, for each entry $w_{i,j}$ of W_{ICA}, it makes a decision about whether $w_{i,j}$ is zero using only $w_{i,j}$.

4.1.1 Deciding which entries are zero

There are several methods for deciding which entries of W_{ICA} to set to zero:

- **Thresholding**: the simplest method for estimating which entries of W_{ICA} are zero is to simply choose a threshold value, and set every entry of W_{ICA} smaller than the threshold (in absolute value) to zero. This method fails to account for the fact that different coefficients may have different spreads, and will miss all coefficients smaller than the threshold.

- **Test the non-zero hypothesis by bootstrap sampling**: another method for estimating which entries of W_{ICA} are actually zero is to do bootstrap sampling. Bootstrap samples are created by resampling with replacement from the original data. Then ICA is run on each bootstrap sample, and each coefficient $w_{i,j}$ is calculated for each bootstrap sample. This leads to a real-valued distribution for each coefficient.[7] Then, for each one, a non-parametric quantile test is performed in order to decide whether 0 is an outlier. If it isn't, the coefficient is set to 0 (i.e. the corresponding edge is pruned.)[8]

- **Use sparse ICA**: Use an ICA algorithm that returns a sparse (i.e. pre-pruned) mixture, such as the one presented by Zhang and Chan [16]. Unlike the other methods above, this is not a local algorithm.

4.1.2 Constrained n-Rooks: the problem and an algorithm

Once it is decided which entries are zero, the algorithm searches for every row-permutation of W_{ICA} that has a zeroless diagonal. Each such row-permutation corresponds to a placement of n rooks onto the non-zero entries on an $n \times n$ chessboard such that no two rooks threaten each other. Then the rows are permuted so that all the rooks end up on the diagonal, thus ensuring that the diagonal has no zeros.

To solve this problem, we use a simple depth-first search that prunes search paths that have nowhere to place the next rook. In the worst case, every permutation is admissible, and the search must take $O(n!)$.

[7] One needs to be careful when doing this, since each run of ICA may return a W_{ICA} in a different row-permutation. This means that we first need to row-permute each bootstrap W_{ICA} to match with the original W_{ICA}.

[8] One could object that, instead of a quantile test, the correct procedure would be to simulate under the null hypothesis (i.e.: edge is absent) using the estimated error terms, and then compare the obtained distribution of the ICA statistics with their distribution for the bootstrap. However, this raises issues and complexities that are tangential to the current paper.

4.2 A non-local algorithm

Local algorithms work under the assumption that the estimates of the $w_{i,j}$ are independent of each other – which is in general false when estimating with finite samples. This motivates the use of non-local methods.

In the LiNGAM (acyclic) approach [12], a non-local algorithm is presented for finding the single best row-permutation of W_{ICA}, which minimizes a loss function that heavily penalizes entries in the diagonal that are close to zero (such as $x \to |1/x|$). This is written as a linear assignment problem (i.e. finding the best match between the $\mathbf{e}_i$s and $\mathbf{x}_i$s), which can be solved using the Hungarian algorithm [9] or others.

For general LiNG discovery, however, algorithms that find the best linear assignment do not suffice, since there may be multiple admissible permutations.

One idea is to use a k-th best assignment algorithm [2] (i.e. the k-th permutation with the least penalty on the diagonal), for increasing k. With enough data, all permutations corresponding to inadmissible models will score poorly, and there should be a clear separation between admissible and inadmissible models.

The non-local method presented above, like the thresholding method, fails to account for differences in spread among estimates of the entries of W_{ICA}. It would be straightforward to fix this by modifying the loss function to penalize diagonal entries for which the test fails to reject the null hypothesis (as described in the part about bootstrap sampling in Section 4.1.1), instead of penalizing them for merely being close to zero.

4.3 Sample run

We generated 15000 sample points using the SEM in Example 1 and error terms distributed according to a symmetric Gaussian-squared distribution[9].

Fig. 3 shows the output of the local thresholding algorithm with the cut-off set to 0.05.

For the sake of reproducibility, our code with instructions is available from: `www.phil.cmu.edu/~tetrad/cd2008.html` .

5 Theory

5.1 Notions of DG equivalence

There are a number of different senses in which the directed graphs associated with SEMs can be "equivalent" or "indistinguishable" given observational data,

[9] The distribution was created by sampling from the standard Gaussian(0,1) and squaring it. If the value sampled was negative, it was made negative again.

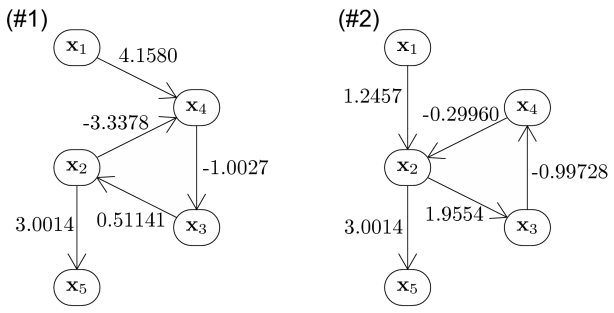


Fig. 3: The output of LiNG-D: Candidate #1 and Candidate #2

assuming linearity and no dependence between error terms:

- DGs G_1 and G_2 are *zero partial correlation equivalent* if and only if the set of zero partial correlations entailed for all values of the free parameters (non-zero linear coefficients, distribution of the error terms) of a linear SEM with DG G_1 is the same as the set of zero partial correlations entailed for all values of the free parameters of a linear SEM with G_2. For linear models, this is the same as *d-separation equivalence*. [13]
- DGs G_1 and G_2 are *covariance equivalent* if and only if for every set of parameter values for the free parameters of a linear SEM with DG G_1, there is a set of parameter values for the free parameters of a linear SEM with DG G_2 such that the two SEMs entail the same covariance matrix over the substantive variables, and vice-versa.
- DGs G_1 and G_2 are *distribution equivalent* if and only if for every set of parameter values for the free parameters of a linear SEM with DG G_1, there is a set of parameter values for the free parameters of a linear SEM with DG G_2 such that the two SEMs entail the same distribution over the substantive variables, and vice-versa. Do not confuse this with the notion of distribution-entailment equivalence between SEMs: two SEMs with fixed parameters are *distribution-entailment equivalent* iff they entail the same distribution.

It follows from well-known theorems about the Gaussian case [13], and some trivial consequences of known results about the non-Gaussian case [12], that the following relationships exist among the different senses of equivalence for acyclic graphs: If all of the error terms are assumed to be Gaussian, distribution equivalence is equivalent to covariance equivalence, which in turn is equivalent to d-separation equivalence. If not all of the error terms are assumed to be Gaussian, then distribution equivalence entails (but is not entailed by) covariance equivalence, which entails (but is not entailed by) d-separation equivalence.

So for example, given Gaussian error terms, $A \leftarrow B$ and $A \rightarrow B$ are zero partial correlation equivalent, covariance equivalent, and distribution equivalent. But given non-Gaussian error terms, $A \leftarrow B$ and $A \rightarrow B$ are zero-partial-correlation equivalent and covariance equivalent, but not distribution equivalent. So for Gaussian errors and this pair of DGs, no algorithm that relies only on observational data can reliably select a unique acyclic graph that fits the population distribution as the correct causal graph without making further assumptions; but for all (or all except one) non-Gaussian errors there will always be a unique acyclic graph that fits the population distribution.

While there are theorems about the case of cyclic graphs and Gaussian errors, we are not aware of any such theorems about cyclic graphs with non-Gaussian errors with respect to distribution equivalence. In the case of cyclic graphs with all Gaussian errors, distribution equivalence is equivalent to covariance equivalence, which entails (but is not entailed by) d-separation equivalence [14]. In the case of cyclic graphs in which at most one error term is non-Gaussian, distribution equivalence entails (but is not entailed by) covariance equivalence, which in turn entails (but is not entailed by) d-separation equivalence. However, given at most one Gaussian error term, the important difference between acyclic graphs and cyclic graphs is that no two different acyclic graphs are distribution equivalent, but there are different cyclic graphs that are distribution equivalent.

Hence, no algorithm that relies only on observational data can reliably select a unique cyclic graph that fits the data as the correct causal graph without making further assumptions. For example, the two cyclic graphs in Fig. 3 are distribution equivalent.

5.2 The output of LiNG-D is correct and as fine as possible

Theorem 1 *The output of LiNG-D is a set of SEMs that comprise a distribution-entailment equivalence class.*

Proof: First, we show that any two SEMs in the output of LiNG-D entail the same distribution.

The weight matrix output by ICA is determined only up to scaling and row permutation. Intuitively, then, permuting the error terms does not change the mixture. Now, more formally:

Let M_1 and M_2 be candidate models output by LiNG-D. Then W_1 and W_2 are row-permutations of W_{ICA}: $W_1 = P_1 W_{ICA}$, $W_2 = P_2 W_{ICA}$

Likewise, for the error terms: $E_1 = P_1 E$, $E_2 = P_2 E$

Then the list of samples X implied by M_1 is $A_1 E_1 = (W_1)^{-1} E_1 = (P_1 W_{ICA})^{-1}(P_1 E) = W_{ICA}^{-1} {P_1}^{-1} P_1 E = W_{ICA}^{-1} E$.

By the same argument, the list of samples X implied by M_2 is also $W_{ICA}^{-1} E$. Therefore, any two SEM models output by LiNG-D entail the same distribution.

Now, it remains to be shown that if LiNG-D outputs one SEM that entails a distribution P, it outputs all SEMs that entail P.

Suppose that there is a SEM S that represents the same distribution as some T, which is output by LiNG-D. Then the reduced-form coefficient matrices for S and T, A_S and A_T, are the same up to column-permutation and scaling. Hence, $I - B_S$ and $I - B_T$ are also the same up to scaling and row-permutation (by $I - B = A^{-1}$). By the assumption that there are no self-loops with coefficient 1, neither $I - B_T$ nor $I - B_S$ has zeros on the diagonal. Since $I - B_T$ is a scaled row-permutation of W_{ICA} that has no zeros on the diagonal, so is $I - B_S$. Thus S is also output by LiNG-D. □

Theorem 2 *If the simultaneous equations are linear and can be represented by a directed graph; the error terms have non-zero variance; the samples are independently and identically distributed; no more than one error term is Gaussian; and the error terms are jointly independent, then in the large sample limit, LiNG-D outputs all SEMs that entail the population distribution.*

Proof: ICA gives pointwise consistent estimates of A and W under the assumptions listed [3]. This entails that there are pointwise consistent tests of whether an entry in the W matrix is zero, and hence (by definition) in the large sample limit, the limit of both type I and type II errors of tests of zero coefficients are zero. Given the correct zeroes in the W matrix, the output of the local version of the LiNG-D algorithm is correct in the sense that the simultaneous equation describes the population distribution. □

In general, each candidate model $B' = I - W'$ has the structure of a row-permutation of W_{ICA}. The structures can be generated by analyzing what happens when we permute the rows of W'. Remember that edges in B' (and thus W') are read column-to-row. Thus, row-permutations of W' change the positions of the arrow-heads (targets), but not the arrow-tails (sources). Richardson proved that the operation of reversing a cycle preserves the set of entailed zero partial correlations, but did not consider distribution equivalence [11].

5.3 Adding the assumption of stability

In dynamical systems, "stable" models are ones in which the effects of one-time noise dissipate. For example, a model that has a single cycle whose cycle-product (product of coefficients of edges in the cycle) is ≥ 1 is unstable, while one that has a single cycle whose cycle-product is between -1 and 1 is stable. On the other hand, if a positive feedback loop of cycle-product 2 is counteracted by a negative loop with cycle-product -1.5, then the model is stable, because the effective cycle-product is 0.5.

A general way to express stability is $\lim_{t \to \infty} B^t = 0$, which is mathematically equivalent to: for all eigenvalues e of B, $|e| < 1$, in which $|z|$ means the modulus of z. This eigenvalues criterion is easy to compute.

Given only the coefficients between different variables, it is impossible to measure the stability of a SEM without assuming something about the self-loops. Therefore, in this section, it is assumed that the true model has no self-loops.

It is often the case that many of the SEMs output by LiNG-D are unstable. Since in many situations, the variables are assumed to be in equilibrium, we are often allowed to rule out unstable models.

In the remainder of this section, we will prove that if the SEM generating the population distribution has a graph in which the cycles are disjoint, then among the candidate SEMs output by LiNG-D, at most one will be stable.

Theorem 3 *SEMs in the form of a simple cycle with a cycle-product π such that $|\pi| \geq 1$ are unstable.*

Proof: Let k be the length of the cycle. Then $B^k = \pi I$. Then for all integers i, $B^{ik} = \pi^i I$. So if $|\pi| \geq 1$, the entries of B^{ik} do not get smaller than the entries of B as i increases. Thus, B^t will not converge to 0 as $t \to \infty$. □

Corollary 1: *For SEMs in the form of a simple cycle, having a cycle-product ≥ 1 is equivalent to having an eigenvalue ≥ 1 (in modulus), which is equivalent to being unstable.*

Theorem 4 *Suppose that there is a SEM M with disjoint cycles with coefficient matrix B and graph G that entails a distribution Q, and a SEM $M_0 \neq M$ with graph G_0, coefficient matrix B_0, which is an admissible permutation of M and also entails Q. Then G_0*

also contains disjoint cycles, at least one of which is a reversal of a cycle C in G, whose cycle-product is the inverse of the cycle-product of C.

Proof: Due to space limitations, the proof is just sketched here. Every permutation can be represented as a product of disjoint cyclic subpermutations of the form $a \to b \to \ldots m \to n \to a$, where $a \to b$ means a gets mapped onto b. (Some cyclic subpermutations may be trivial, i.e. contain a single object mapped onto itself). Hence it suffices to prove the theorem for a single admissible cyclic row permutation of B. It can be shown that if a cyclic row permutation of B, $a \to b \to \ldots m \to n \to a$ is admissible, then G contains the cycle C equal to $a \leftarrow b \leftarrow \ldots m \leftarrow n \leftarrow a$, and G_0 contains the reversed cycle C equal to $a \to b \to \ldots m \to n \to a$. Moreover, if G_0 contains two cycles that touch, so does G.

Consider B_C, the submatrix of B that contains the coefficients of the edges in cycle C.

$$B_C = \begin{bmatrix} 0 & \ldots & 0 & b_{k,1} \\ b_{1,2} & 0 & \ldots & 0 \\ 0 & b_{2,3} & \ddots & 0 \\ 0 & 0 & \ddots & 0 \end{bmatrix}$$

Note that the cycle-product $\pi_C = b_{k,1} \prod_{i=0}^{k-1} b_{i,i+1}$.

$W_C = I - B_C$.

The "reversal" is the row-permutation in which the first row gets "rotated" into the bottom:

$$RowPermute(W_C) = \begin{bmatrix} -b_{1,2} & 1 & \ldots & 0 \\ 0 & -b_{2,3} & \ddots & 0 \\ 0 & 0 & \ddots & 1 \\ 1 & 0 & \ldots & -b_{k,1} \end{bmatrix}$$

Normalizing the diagonal to be all 1s, we get $W_{C'}$. Computing $B_{C'} = I - W_{C'}$, one can see that the cycle-product $\pi_{C'} = \frac{1}{b_{k,1}} \prod_{i=0}^{k-1} \frac{1}{b_{i,i+1}} = 1/\pi_C$. □

We will now show that for SEMs in which the cycles are disjoint, their stability only depends on the stability of the cycles.

Theorem 5 *A SEM in which the cycles are disjoint is stable if and only if it has no unstable cycles.*

Proof: Let be G be a SEM whose cycles are disjoint. Then B_G can be written as a block-triangular matrix where each diagonal block is a cycle. The set of eigenvalues of a block-triangular matrix is the union of the sets of eigenvalues of the blocks in the diagonal (in this case, the eigenvalues of the cycles). Suppose a cycle of G is unstable. Then it has an eigenvalue ≥ 1 (in modulus). But since this is also an eigenvalue of B_G, it follows that G is unstable. The other direction goes similarly. □

Theorem 6 *If the true SEM is stable and has a graph in which the cycles are disjoint, then no other SEMs in the output of LiNG-D will be stable.*

Proof: Suppose the true SEM is stable and has a graph in which the cycles are disjoint. Call it G. Since, by Theorem 2, the output of LiNG-D are the admissible distribution-entailment equivalent alternatives to the true SEM, it suffices to show that all other admissible candidates are unstable.

By Theorem 5, all cycles in G are stable. Let H be an admissible alternative to G, such that $H \neq G$. By Theorem 4, H will have at least one cycle C reversed relative to G and this reversed cycle will have a cycle product that is the inverse of the cycle product of C. By Corollary 1, the reversed matrix is not stable. Thus, by Theorem 5, H is unstable.

Therefore, the only stable admissible alternative to G is G itself. □

It follows that if the true model's cycles are disjoint, then under the assumption that the true model is stable, we can fully identify it using a LiNG discovery algorithm (at most one SEM in the output of the LiNG discovery algorithm will be stable).

For example, consider the two candidate models shown in Fig. 3. By assuming that the true model is stable, one would select candidate #2. Since our simulation used a stable model, this is indeed the correct answer (see Fig. 1).

In general, however, there may be multiple stable models, and one cannot reliably select the correct one. When the cycles are not disjoint, it is easy to find examples for which there are multiple stable candidates.

The condition of disjoint cycles is sufficient, but not necessary: it is easy to come up with SEMs where we have exactly one stable SEM in the distribution-entailment equivalence class, despite intersecting cycles.

6 Discussion

We have presented Shimizu's approach for discovering LiNGAM SEMs, and generalized it to a method that discovers general LiNG SEMs. This improves upon the state-of-the-art on cyclic linear SEM discovery by outputting only the distribution-entailment equivalence class of SEMs, instead of the entire d-separation equiv-

alence class; and by relaxing the faithfulness assumption. We have also shown that stability can be a powerful constraint, sometimes narrowing the candidates to a single SEM.

There are a number of questions that remain open for future research:

- The LiNG-D algorithm generates all admissible permutations. The worst-case time-complexity of n-Rooks is high, but can we do better than depth-first search for random instances? Is there an algorithm to efficiently search for the stable models, without going through all candidates? In the case where the cycles are disjoint, it is possible to just find the correct permutation for each cycle independently, but no such trick is known in general.

- How can prior information be incorporated into the algorithm?

- How can the algorithm be modified to allow the assumption of causal sufficiency to be relaxed? For the acyclic case, see [7].

- How can the algorithm be modified to allow for mixtures of non-Gaussian and Gaussian (or almost Gaussian) error terms? Hoyer et al [6] address this problem for the acyclic case.

- How could we integrate this method into mainstream dynamical systems research? Can the algorithm handle noisy dynamics and noisy observations? Could it be made to handle non-linear dynamics? What about self-loops of coefficient 1? How could one integrate this with methods that use non-equilibrium time-series data?

Acknowledgements

The authors wish to thank Anupam Gupta, Michael Dinitz and Cosma Shalizi. GL was partially supported by NSF Award No. REC-0537198.

References

[1] K. Bollen (1989) - *Structural Equations with Latent Variables*, John Wiley & Sons, New York.

[2] C. R. Chegireddy, H. W. Hamacher (1987) - Algorithms for finding K-best perfect matchings *Discrete Applied Mathematics*, **18**:155-165.

[3] P. Comon (1994) - Independent component analysis - a new concept? *Signal Processing*, **36**:287-314.

[4] D. Dash (2005) - Restructuring Dynamic Causal Systems in Equilibrium. *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AIStats 2005)*

[5] F. Fisher (1970) - A correspondence principle for simultaneous equation models. *Econometrica,* **38**(**1**)*:73-92.*

[6] P. O. Hoyer, A. Hyvärinen, R. Scheines, P. Spirtes, J. Ramsey, G. Lacerda, S. Shimizu (2008) - Causal discovery of linear acyclic models with arbitrary distributions. *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence.*

[7] P. O. Hoyer, S. Shimizu, and A. J. Kerminen (2006) - Estimation of linear, non-gaussian causal models in the presence of confounding latent variables *Proc. Third European Workshop on Probabilistic Graphical Models (PGM'06), pp. 155-162*

[8] A. Hyvärinen, J. Karhunen, E. Oja (2001) - *Independent Component Analysis.* Wiley Interscience.

[9] H. W. Kuhn (1955) - The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, **2**:83-97, 1955.

[10] J. Pearl (2000) - *Causality: Models, Reasoning, and Inference.* Cambridge University Press.

[11] T. Richardson (1996) - A Polynomial-Time Algorithm for Deciding Markov Equivalence of Directed Cyclic Graphical Models. *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*

[12] S. Shimizu, P. Hoyer, A. Hyvärinen, A. Kerminen (2006) - A linear, non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research* **7**:2003-2030.

[13] P. Spirtes, C. Glymour, R. Scheines (1993) - *Causation, Prediction, Search.* Springer-Verlag Lecture Notes in Statistics 81.

[14] P. Spirtes (1995) - Directed Cyclic Graphical Representation of Feedback Models, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*

[15] R. Strotz and H. Wold (1960) - Recursive versus nonrecursive systems: an attempt at synthesis. *Econometrica* **28**:417-427.

[16] K. Zhang, L-W. Chan (2006) - ICA with Sparse Connections *Intelligent Data Engineering and Automated Learning IDEAL 2006.*

Improving Gradient Estimation by Incorporating Sensor Data

Gregory Lawrence
Computer Science Division
U.C. Berkeley
gregl@cs.berkeley.edu

Stuart Russell
Computer Science Division
U.C. Berkeley
russell@cs.berkeley.edu

Abstract

An efficient policy search algorithm should estimate the local gradient of the objective function, with respect to the policy parameters, from as few trials as possible. Whereas most policy search methods estimate this gradient by observing the *rewards* obtained during policy trials, we show, both theoretically and empirically, that taking into account the *sensor data* as well gives better gradient estimates and hence faster learning. The reason is that rewards obtained during policy execution vary from trial to trial due to noise in the environment; sensor data, which correlates with the noise, can be used to partially correct for this variation, resulting in an estimator with lower variance.

1 INTRODUCTION

Policy search algorithms have been very effective in learning good policies in the reinforcement learning setting. Successful applications include helicopter flight [8], quadruped locomotion [5], and baseball hitting [10]. These methods work by adjusting the parameters of a policy to improve its *value*, i.e., the expected sum of rewards (possibly discounted) obtained during policy execution. To do this, the algorithms repeatedly estimate the *gradient* of the value with respect to the parameters, using information observed during policy trials, and then adjust the parameters in the "uphill" direction. Because trials can be expensive, especially in physical environments, a number of authors have presented techniques to reduce the number of required trials—mainly by reducing the variance of the gradient estimator [1, 3, 6, 7, 10, 9, 11, 13].

Generally speaking, these methods estimate the gradient from the policy parameter settings on each trial and the *score* (the actual sum of rewards), ignoring the sensor data.[1] The main point of this paper is that the sensor data obtained during each trial also provides a useful signal that can reduce the variance of gradient estimators. To understand how this may be so, consider first a case in which it is *not* so: that is, the noise-free case where the score is a deterministic function of the policy parameters. In that case, the local gradient can be estimated exactly from a small set of trials with policy parameter settings closely spaced around the setting of interest, and the sensor data can provide no more information.[2] In the noisy case, however, a gradient estimator can be easily misled by trials of bad policies that yield fortuitously good scores and *vice versa*. In essence, what we propose is that sensor data can account, at least partially, for the *deviation* in the score of each trial from its expected value. Conditioned on the sensor data, therefore, the posterior estimate for the policy value will be closer to the true value.

As an example, consider the problem of firing a cannon at a distant target (Figure 1). Imagine that, after firing several shots that fall short, you increase the firing angle to $\theta = \theta_0$. The next shot sails over the target. Normally, you would decrease θ again. Suppose, however, that the sound of this last shot was much louder than usual; this suggests that the muzzle velocity was higher than intended and accounts for the poor outcome. It might therefore be sensible to stay at θ_0 for the next shot. Notice, though, that this decision re-

[1] Of course, the policies themselves may use the sensor data to select actions. One might also interpret the approach of [6] as using observed perturbations to improve gradient estimation—but only for the restricted case of perfect sensing of noise applied directly to the policy parameters. The current paper removes these unrealistic assumptions, allowing the method to apply to any real physical system; it also provides a more general explanation of the benefits of sensor data for gradient estimation.

[2] If we are willing to step outside the policy search framework, of course, then we can use the sensor data to learn a transition model from which the optimal policy can be obtained by offline methods.

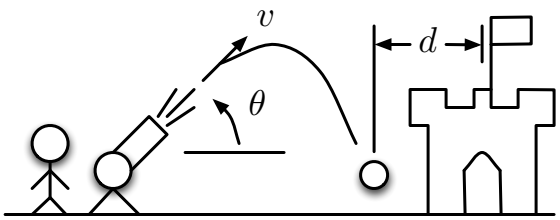


Figure 1: The cannon problem.

quires some knowledge of the relationship between the sensor data (loudness of bang) and the score; learning such relationships is a key element of our method.

The remainder of the paper is organized as follows. Section 2 introduces the basic technical approach: instead of estimating the gradient by fitting a local linear model for the value as a function of the policy parameters, we fit a linear model of the value as a function of the policy parameters and the (transformed) sensor values. We show that estimator variance is minimized if the sensor data is transformed by a projection that renders it statistically independent of the policy parameters while remaining correlated with the perturbations in the score; we also present a practical technique that approximates this ideal transformation. Section 3 describes how we applied these techniques to a dart throwing task and a quadruped locomotion task, in both cases obtaining improved learning curves compared to a method that does not use the sensor data. Section 4 discusses possibilities for future improvements.

2 INCORPORATING SENSOR INFORMATION

A policy π determines how an agent chooses its actions given its past observations, and the reinforcement learning goal is to find a policy π^* that maximizes the performance. Each policy trial generates a *history* h, a sequence of percept-action-reward tuples; the *score* $f(h)$ is the sum of the reward values. The optimal policy π^* maximizes the value function $V(\pi) = \mathrm{E}[f(\mathbf{h})|\pi]$ where the histories $\mathbf{h}$ are generated from π. In this paper we perform policy search by hill-climbing through a space of parameterized policies $\pi \in \mathbb{R}^d$. We climb through this space by adjusting our current policy π_0 in the direction of the gradient $\nabla_\pi V(\pi)|_{\pi=\pi_0}$. At each hill-climbing step, the gradient is estimated from n policy trials where we vary the parameters of each trial for exploration purposes.

2.1 TOY EXAMPLE

To illustrate the main contributions of this paper, we will examine the toy cannon problem (Figure 1). Here, the policy $\pi = (\pi_v, \pi_\theta)^T$ consists of a desired cannon angle, $\pi_\theta \in [0, \pi/2]$ and a desired initial velocity $\pi_v > 0$. Following [6], we assume the policy itself is perturbed by noise to give the actual controls $u = (u_v, u_\theta)^T$. We assume that the agent has access to a noisy sensor that measures the perturbation $(u - \pi)$ and let $s = (s_v, s_\theta)$ denote its value. There is additive, zero-mean Gaussian noise in both the control and sensor values; the control noise has covariance matrix Σ_u and the sensor noise has covariance matrix Σ_s. The history h for this problem has a single tuple with the desired action π, the sensor value s, and the reward $-d^2$, where d is the distance from the target to where the cannon ball lands. The score is just the reward in this case, so maximizing $V(\pi) = \mathrm{E}[f(\mathbf{h})|\pi]$ is equivalent to minimizing the expected squared distance error.

To demonstrate the benefits of incorporating the sensor data, suppose that we had a perfect sensor. In that case, we have $u = \pi + s$. Furthermore let us assume that $f(h)$ is (locally) linear in the actual control u. Then the score function can be written as $f(\pi, s) = (\pi + s)^T A_{\pi_0} + b_{\pi_0}$. A typical approach to gradient estimation in this setting would be to ignore the sensor data and use samples of π and $f(\pi, s)$ to fit a local linear approximation to V around π_0:

$$\hat{V}(\pi) = \pi^T A_{\pi_0} + b_{\pi_0} \ .$$

Using least-squares regression, we estimate the gradient by learning A_{π_0} from n policy trials. The control noise causes perturbations in the scores with variance given by ${A_{\pi_0}}^T \Sigma_u A_{\pi_0}$. Hence, the estimator A_{π_0} will not be exact and we may need many samples to be confident in our estimate of the gradient.

To obtain the benefits of incorporating the sensor data, we can instead fit a sensor-data-dependent linear approximation to the scoring function f itself (rather than its expectation V):

$$\hat{f}(\pi, s) = \pi^T A_{\pi_0} + s^T A_s + b_{\pi_0} \ .$$

From samples of π, $f(\pi, s)$, and s we can use linear regression to learn both A_{π_0} and A_s in this equation. The value that we learn for A_{π_0} can be used as our estimate of the gradient of V. In the case of perfect sensing, $\hat{f}(\pi, s)$ can be a locally *exact* fit because the score will be a deterministic function of π and s. Thus, we have a perfect gradient estimator. Intuitively, the more informative the sensor, the better our gradient estimator.

There are two important issues to note with this analysis. As we will see in Section 2.2, the estimator A_{π_0}

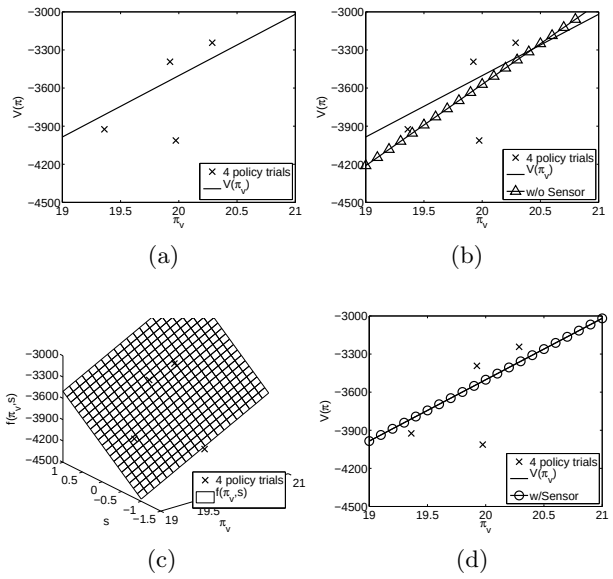


Figure 2: (a) The true value function with four policy trials superimposed. (b) Least squares fit of these four trials. (c) Least squares fit of the the four trials as a function of π and s. (d) Projection of the fitted plane from (c) to give a linear estimator for $V(\pi)$ (circles). Note that the estimator fits the true value function exactly.

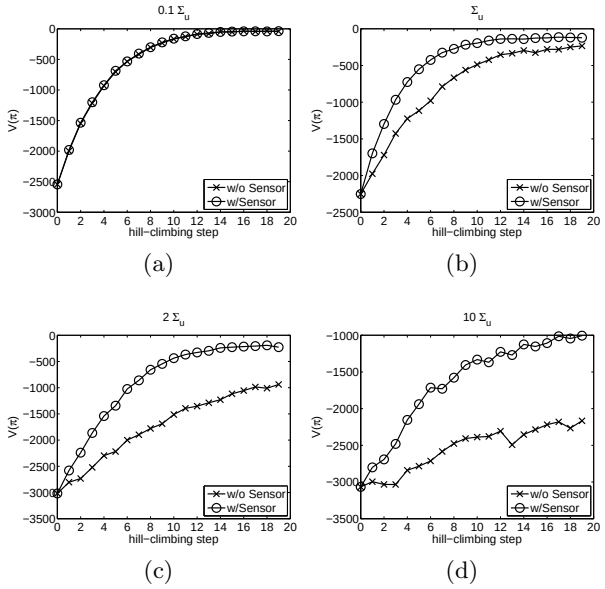


Figure 3: The learning curve performances of two difference policy search algorithms as we increase the level of actuator noise. One curve estimates the gradient while ignoring the sensor data and the other curve estimates the gradient using the sensor data. As mentioned in Section 1, using the sensor data becomes more beneficial as the level of noise in the score increases.

will be unbiased *provided* the sensor data are independent of the policy parameters. Therefore, it will be advantageous to project the sensor data in such a way as to render it independent of the policy parameters to the extent possible, while maintaining its correlation with the perturbations in the score. Second, the sensor-data-dependent fit for $\hat{f}$ requires extra parameters to be learned for A_s (two extra parameters in our example), which may in turn require extra trials. We will see in the next subsection that the relative efficiency of this estimator depends on the amount of noise in the observed scores that can be corrected for using the sensor data and the number of extra parameters to be learned.

Figures 2(a-d) demonstrates how incorporating the sensor data is beneficial. For ease of graphical depiction, we consider a restricted version of the cannon problem in which u_θ is fixed at 45 degrees. We consider a linear approximation to the score around the nominal cannon velocity $\pi_{v0} = 20\text{ms}^{-1}$. Figure 2(a) shows the true value function with four policy trials superimposed. Figure 2(b) shows the least squares fit of these four trials. The slope of this line is an estimate of the gradient of $V(\pi)$. Figure 2(c) shows the least squares fit of the four trials as a function of π and s and Figure 2(d) shows this least squares fit projected onto the plane that spans the π_v and $f(h)$ axes. Notice that it perfectly fits the true value. This is because deviations from the expected score are explained away by the sensor values. We expand upon this in the next section.

Figure 3 shows different learning curves for the cannon problem as we increase the level of actuator noise ($\Sigma_u = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 4 \end{smallmatrix}\right]$). At each hill-climbing step we ran 10 trials, each with a different policy in the neighborhood of the current policy, and we averaged the resulting learning curve over 100 hill-climbing runs. One curve shows the learning performance achieved while ignoring the sensor data while the other curve shows what happens when incorporating the sensor data. Notice that the performance gains from using sensor data become more pronounced as the noise level increases.

2.2 VARIANCE REDUCTION

In this section we compare the variance of a gradient estimator that incorporates sensor data to an estimator that ignores it. We assume that the score function is an unknown linear function of both the policy parameters and the sensor data. The gradient estimators take n policy trials from the current hill-climbing step and return an estimate of $\nabla_\pi V(\pi)|_{\pi=\pi_0}$.

We will show that an agent can reduce the variance of its gradient estimates by choosing a sensor encoding that correlates with the noise-induced perturbations in the score. To get an unbiased estimate, the sensor data must be uncorrelated with π. In Section 2.2.3 we

give expressions for the bias and variance of a gradient estimator in the correlated setting. A biased estimator that incorporates sensor data may still outperform one that ignores it as long as the bias remains small.

Let $\Pi = [\pi_1, \ldots, \pi_n]^T$ be the matrix of policy parameters and let $S = [s_1, \ldots, s_n]^T$ be a matrix whose rows contain the corresponding sensor values. We assume that the sensors have a Gaussian distribution with variance Σ_s and mean μ_s. Let $f = [f_1, \ldots, f_n]^T$ be a column vector whose entries are the score values and let $w = [w_1, \ldots, w_n]$ be a column vector of zero-mean noise which is added to the output with variance ${\sigma_w}^2$. The score is written as $f_2(\pi, s, w) = \pi^T A_\pi + s^T A_s + b + w$. The scores for the current set of policy trials are given by the following equation:

$$f = \Pi A_\pi + S A_s + \mathbf{1}_n b + w, \tag{1}$$

where $\mathbf{1}_n$ is a column vector of ones.

Each estimator learns a local linear model of the scoring function using linear regression on the data obtained from n policy trials. For exploration purposes the policy parameters of each trial are assumed to be distributed around a nominal policy π_0 according to a Gaussian distribution with variance Σ_e. We assume that the policy parameters, scores, and sensor data have been centered around zero.

2.2.1 Ignore Sensor Data

From the point of view of a gradient estimator that ignores the sensor data, additional noise will appear to be added to the scores that will not be explained by perturbations in the sensor data. Let v represent this noise where each element is given by the equation $v = s^T A_s + w$. The variance of each entry in v is given by the expression ${A_s}^T \Sigma_s A_s + \sigma^2$. The score function $f_1(\pi, v) = \pi^T A_\pi + b + v$ is equivalent in value to f_2. We can learn the linear relationship between the policy parameters and the score by performing linear regression on the set of n policy trials. In other words we find a suitable estimate for A_π in the following equation:

$$f = \Pi A_\pi + \mathbf{1}_n b + v. \tag{2}$$

We are interested in the gradient of $\mathrm{E}[f_1(\boldsymbol{\pi}, \mathbf{v})|\pi_0]$, where $\boldsymbol{\pi}$ and $\mathbf{v}$ represent random variables whose values are distributed according to the exploration distribution $\mathcal{N}(\pi_0, \Sigma_e)$ and the output distribution respectively. The gradient with respect to π_0 is written as $\nabla_{\pi_0} \mathrm{E}[f_1(\boldsymbol{\pi}, \mathbf{v})|\pi_0] = A_\pi$ and therefore, a sensible gradient estimator returns an estimate of A_π from the n policy trials. The gradient estimator, which we denote by $g_1(\Pi, f)$ is given by the linear regression equation:

$$g_1(\Pi, f) = (\Pi^T \Pi)^{-1} \Pi^T f$$

The variance of the g_1 is written as

$$\begin{aligned}\mathrm{var}[g_1(\Pi, \mathbf{f})] &= (\Pi^T \Pi)^{-1} \Pi^T \mathrm{E}[\mathbf{v}\mathbf{v}^T] \Pi (\Pi^T \Pi)^{-1} \\ &= (\Pi^T \Pi)^{-1} \Pi^T ({A_s}^T \Sigma_s A_s + \sigma^2) \Pi (\Pi^T \Pi)^{-1} \\ &= (\Pi^T \Pi)^{-1} ({A_s}^T \Sigma_s A_s + \sigma^2),\end{aligned}$$

where $\mathbf{f}$ is a column vector random variable whose entries are distributed according to the output distribution. This quantity is for a fixed set of policies Π. The variance of the g_1 averaged over the randomness of the policies drawn for exploration purposes can be determined by observing that the distribution of the matrix $(\Pi^T \Pi)^{-1}$ is an inverted Wishart with n degrees of freedom where d is the number of policy parameters.

$$\mathrm{var}[g_1(\boldsymbol{\Pi}, \mathbf{f})] = \frac{{\Sigma_e}^{-1} ({A_s}^T \Sigma_s A_s + \sigma^2)}{(N - d - 1)}, \tag{3}$$

where $\boldsymbol{\Pi}$ is a random variable where each row is distributed according to the exploration distribution.

2.2.2 Include Sensor Data

A linear model that predicts the score as a function of both the policy parameters and sensor data will have the noise on the output partially explained by the sensor data. An estimate of the score as a linear function of the policy parameters and sensor data can be written as

$$\begin{bmatrix} g_2(\Pi, S, f) \\ \beta_2(\Pi, S, f) \end{bmatrix} = \begin{bmatrix} \Pi^T \Pi & \Pi^T S \\ S^T \Pi & S^T S \end{bmatrix}^{-1} \begin{bmatrix} \Pi^T f \\ S^T f \end{bmatrix},$$

where $\beta_2(\Pi, S, f)$ determines how the sensor values affect the perturbations in the score. The gradient of $\mathrm{E}[f_2(\boldsymbol{\pi}, \mathbf{s}, \mathbf{f})|\pi_0]$ with respect to π_0 is written as $\nabla_\pi \mathrm{E}[f_2(\boldsymbol{\pi}, \mathbf{s}, \mathbf{f})|\pi_0] = A_\pi$ and so we can use $g_2(\Pi, S, f)$, the first d entries in the above vector, as our estimate of the gradient. The variance of the above expression is written as

$$\mathrm{var} \begin{bmatrix} g_2(\Pi, S, \mathbf{f}) \\ \beta_2(\Pi, S, \mathbf{f}) \end{bmatrix} = \begin{bmatrix} \Pi^T \Pi & \Pi^T S \\ S^T \Pi & S^T S \end{bmatrix}^{-1} \sigma^2.$$

We take the inverse of the Schur complement with respect to $S^T S$ to find the variance of g_2:

$$\mathrm{var}[g_2(\Pi, S, \mathbf{f})] = (\Pi^T \Pi - \Pi^T S (S^T S)^{-1} S^T \Pi)^{-1} \sigma^2.$$

This quantity is for a fixed set of policies Π and sensor values S. The variance of g_2 averaged over different exploration policies and sensor values, assuming that the sensors are independent of both the policy parameters and output noise w, is given by the following equation:

$$\begin{aligned}\mathrm{var} \begin{bmatrix} g_2(\boldsymbol{\Pi}, \mathbf{S}, \mathbf{f}) \\ \beta_2(\boldsymbol{\Pi}, \mathbf{S}, \mathbf{f}) \end{bmatrix} &= \begin{bmatrix} \Sigma_e & 0 \\ 0 & \Sigma_s \end{bmatrix}^{-1} \frac{\sigma^2}{(N - d - d_s - 1)} \\ \mathrm{var}[g_2(\boldsymbol{\Pi}, \mathbf{S}, \mathbf{f})] &= \frac{{\Sigma_e}^{-1} \sigma^2}{(N - d - d_s - 1)},\end{aligned} \tag{4}$$

where d_s is the dimensionality of the sensor data.

The expressions for the variance of the two gradient estimators (equations 3 and 4) differ from each other in two factors. The variance of the estimator that ignores the sensor data has a factor of $({A_s}^T \Sigma_s A_s + \sigma^2)$ which is reduced to σ^2 in the estimator that incorporates the sensor data. We see that we get bigger reductions whenever the sensor information provides more information about the score. The second difference between the two estimators favors the estimator that ignores the sensor data because the denominator in equation 3 has a term that is larger than the corresponding term in equation 4. The difference in the denominators is the dimensionality of the sensor data d_s, which suggests that we should choose sensor encodings of low dimensionality. Whether g_2 is more efficient than g_1 depends on the relative strength of these two factors.

2.2.3 Correlated Sensors

If the sensors are correlated with the policy parameters then the gradient estimator that incorporates sensor data will be biased. In this situation we can represent the distribution over sensors as a linear Gaussian distribution $s \sim \mathcal{N}({A_{\pi,s}}^T \pi + b_{\pi,s}, \Sigma_s)$. Inserting this into score function f_2 gives the following equation:

$$f_3(\pi, s, w, w_{\pi,s}) = \pi^T A_\pi + \pi^T A_{\pi,s} A_s + {b_{\pi,s}}^T A_s + {w_{\pi,s}}^T A_s + b + w$$

where $w_{\pi,s}$ is a zero-mean Gaussian random variable with variance Σ_s. The gradient with respect to π_0 is written as $\nabla_{\pi_0} \mathrm{E}[f_3(\boldsymbol{\pi}, \mathbf{s}, \mathbf{w}, \boldsymbol{w_{\pi,s}})|\pi_0] = A_\pi + A_{\pi,s} A_s$ which means that gradient estimator g_2 is biased by $A_{\pi,s} A_s$ whenever the sensors are correlated with the policy parameters. The variance of the estimator also changes in the case of correlated sensors:

$$\mathrm{var}[g_2(\boldsymbol{\Pi}, \mathbf{S}, \mathbf{f})] = \frac{(\Sigma_e - D)^{-1}\sigma^2}{(N - d - d_s - 1)}$$
$$D := \Sigma_{es}({A_{\pi,s}}^T \Sigma_e A_{\pi,s} + \Sigma_s)^{-1} {\Sigma_{es}}^T,$$

where $\Sigma_{es} = \Sigma_e A_{\pi,s}$ is the covariance of the policy parameters and sensor values. Thus we see that it is best to choose sensor encodings where the sensor values are uncorrelated with the policy parameters.

2.3 FINDING GOOD SENSOR ENCODINGS

In the prior subsection we saw that S should be independent of Π to give an unbiased estimate of the gradient. This is often not true in many problems. The analyzes also suggests that to get an improved gradient estimator, we should prefer low-dimensional sensor encodings that have a great influence on the score. This section presents a heuristic that can be used to find good sensor encodings that give performance gains in problems where the assumptions do not hold.

We find good sensor encodings by taking sensor variables that lie in a high dimensional space and projecting them down to a low-dimensional subspace. Intuitively, we should prefer directions that maintain the influence of the sensor data on the score. We also want the sensor data to be uncorrelated with the policy parameters to minimize the bias. Our approach is to search over possible sensor encodings to find the optimal projection at each hill-climbing step. We use cross-validation to measure the quality of each projection. Let $\Pi_{\text{-}i}$ equal Π with the ith row removed, let $S_{\text{-}i}$ equal S with the ith row removed, and let $f_{\text{-}i}$ equal f with the ith entry removed. Let B be a matrix that projects the raw sensor data down to a low-dimensional subspace. We use a quasi-Newton method to minimize the following cost function:

$$J = \sum_{i=1}^{n} ({\pi_i}^T g(\Pi_{\text{-}i}, S_{\text{-}i} B, f_{\text{-}i}) + o(\Pi_{\text{-}i}, S_{\text{-}i} B, f_{\text{-}i}) - f_i)^2,$$

where g estimates the gradient using g_2 after the inputs have been centered around zero and o estimates the appropriate offset term. Thus we learn the gradient and corresponding offset terms using the data from a collection of policy trials with a single policy trial held out at a time. This gradient and offset is then used to predict the score of the held out sample. Given the optimal projection B^* from the above procedure, we estimate the gradient using the full set of policy trials:

$$\nabla_\pi V(\pi)|_{\pi=\pi_0} \approx g_2(\Pi, SB^*, f). \tag{5}$$

3 RESULTS

In this section we describe how an agent can use its sensor data to improve its learning performance on a dart throwing task [6] and a quadruped locomotion task. These partially observable sequential tasks are complicated partly due to the fact that the transition and sensor models and their structures are unknown. These properties are characteristic of a wide range of real-world tasks.

Figure 4(a) shows a single frame of the dart throwing problem where the objective is to throw a dart with minimal mean squared error (measured from where the dart hits the wall to the center of the dart board). The arm is modeled as a three-link rigid body with dimensions based on biological measurements [2]. The links correspond to the upper arm, forearm, and hand and are connected using a single degree of freedom

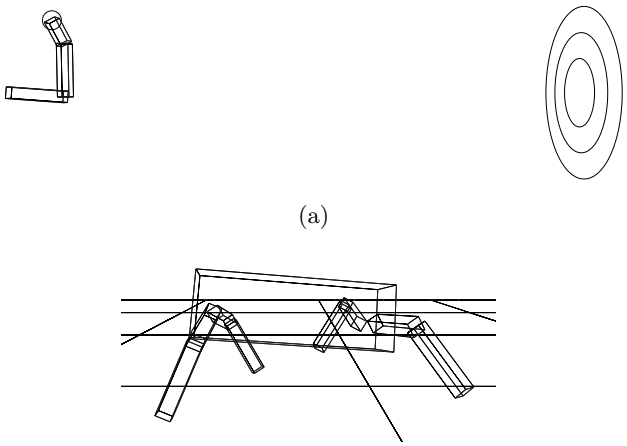


Figure 4: (a) Dart throwing problem (b) Quadruped locomotion problem.

rotational joint. The upper arm is connected to the shoulder at a fixed location. The arm is controlled by applying torques at each joint. These torques are generated by a PD-controller that attempts to move the arm through a desired trajectory, specified by a cubic spline for each joint angle. The starting posture of the arm is fixed in advance and the control policy specifies three spline knot positions for each joint, yielding 9 policy parameters in all. The controller is simulated for approximately 0.2 seconds and then the dart is released. The torques calculated by the PD-controller are perturbed by additive and multiplicative noise,[3] and there is Gaussian noise added to the release time with $\sigma = 0.01$.

Figure 4(b) shows a single frame of the quadruped locomotion problem where the objective is to maximize the sustained speed of the robot. Each leg of the quadruped has four degrees of freedom (three at the shoulder joints and one at the elbow). The quadruped is controlled by applying torques at each joint which gives the system 16 degrees of controllable freedom. The torques are generated by a PD-controller that attempts to move each leg through a desired trajectory, specified by a truncated Fourier Series for each joint angle. Each controllable degree of freedom has three corresponding parameters and the right side of the body is constrained to move the same as the left except offset by 180 degrees; there are 24 policy parameters. The controller is simulated for 3 seconds for each policy trial and the distance travelled is the objective function. As with the dart thrower, noise enters the system by perturbing the torques given by the PD-controller by additive and multiplicative noise.

[3]Multiplicative noise has been shown to explain some aspects of biological motion [4, 12].

3.1 SENSORS FOR MOTOR CONTROL

The systems described in the previous subsection are capable of measuring the state of the observable joint angles during each policy trial. In these two tasks, the observable joints are the same as the controllable joints. In the quadruped task, this means that while the agent can sense the positions of each leg relative to the body, it does not have access to the absolute position and rotation of the torso.

Our task is to take the sensed trajectories and to transform the values to something that gives us improvements in our gradient estimator. The sensor encodings should be independent of the policy parameters and so we attempt to find the difference between the observed motion of the system and the expected motion at each time step. The idea of using sensory data to cancel out the effect of one's own motion is also present in the biology literature [14]. We approximate this difference by learning a crude approximation to the dynamical system in a pre-processing phase. Using the joint-space representation of each system, the dynamics are governed by the following second-order nonlinear differential equation:

$$m(x)\ddot{x} = u(t) + g(x) + c(x, \dot{x}) + w(x, \dot{x}, t).$$

where x is the physical state of the system, $m(x)$ is the joint-space inertia matrix, $u(t)$ are the forces and torques applied to the system, $g(x)$ is the gravitational force, $c(x, \dot{x})$ are the Centrifugal and Coriolis forces, and $w(t)$ is the noise plus any external forces (e.g., the ground pushing up on the feet of the quadruped). A discrete time version of this equation is written as:

$$m(x)a = u(t) + g(x) + c(x, v) + w(x, v, t).$$

where v are the velocities and a is the acceleration.

We approximate the expected acceleration by predicting the following quantities as a function of the observable states and velocities:

$$\begin{aligned} \text{vec}(m(x)^{-1}) &\approx A_M \phi(x_{(o)}) \\ \text{vec}(m(x)^{-1} g(x)) &\approx A_G \phi(x_{(o)}) \\ \text{vec}(m(x)^{-1} c(x, v)) &\approx A_C \phi(\left[{x_{(o)}}^T, {v_{(o)}}^T\right]^T), \end{aligned}$$

where $x_{(o)}$ and $v_{(o)}$ contain the observable components of the state and velocity terms and where $\phi(x) = \text{triu}([1, x^T]^T[1, x^T])$ is a function that augments its input with quadratic terms ($\text{triu}(X)$ is a function that returns the upper triangular part of X stacked as a single column vector). We use linear regression to learn a model of each of these components as a function of the observable state variables. For example in the quadruped problem we will learn these linear models

without regard to the absolute rotation of the system. This is clearly an approximation for the terms that involve gravity because the direction of the gravitational force, from a frame of reference attached to the torso, depends on its rotation relative to the ground frame.

We learn these parameters in a pre-processing stage by examining random states (x_i, v_i) in the dynamical system and examining the mass matrix $m(x_i)$, the gravity forces $g(x_i)$, and the Coriolis and Centrifugal forces $c(x_i, v_i)$. The samples are drawn from a distribution of states that are likely to be encountered during policy execution. We learn the linear relationships using the following equations:

$$A_M := (\Phi_1{}^T \Phi_1)^{-1} \Phi_1^T M$$
$$A_G := (\Phi_1{}^T \Phi_1)^{-1} \Phi_1^T G$$
$$A_g := (\Phi_2{}^T \Phi_2)^{-1} \Phi_2^T C,$$

where

$$\begin{aligned}
\alpha_i &= \phi(x_{i(o)}) \\
\beta_i &= \phi([x_{i(o)}{}^T, v_{i(o)}{}^T]^T) \\
\Phi_1 &= [\alpha_1{}^T, \ldots, \alpha_{n_f}{}^T]^T \\
\Phi_2 &= [\beta_1{}^T, \ldots, \beta_{n_f}{}^T]^T \\
M &= [\mathrm{vec}(m(x_1)^{-1})^T, \ldots, \mathrm{vec}(m(x_{n_f})^{-1})^T]^T \\
G &= [\mathrm{vec}(m(x_1)^{-1} g(x_1))^T, \ldots, \\
&\quad \mathrm{vec}(m(x_{n_f})^{-1} g(x_{n_f}))^T]^T \\
C &= [\mathrm{vec}(m(x_1)^{-1} c(x_1, v_1))^T, \ldots, \\
&\quad \mathrm{vec}(m(x_{n_f})^{-1} c(x_{n_f}, v_{n_f}))^T]^T.
\end{aligned}$$

During each policy execution we can take these parameters to estimate the expected acceleration at each time step as follows:

$$\begin{aligned}
\hat{a}(\pi, t, x_{(o)}, v_{(o)}) = \; & \mathrm{resh}_M(A_M \phi(x_{(o)})) u(\pi, t) + \\
& \mathrm{resh}_G(A_G \phi(x_{(o)})) + \\
& \mathrm{resh}_C(A_C \phi([x_{(o)}{}^T, v_{(o)}{}^T]^T)),
\end{aligned}$$

where the resh function reshapes a matrix to its original size. The difference in velocity is computed as the actual velocity at each time step $v_{(o)}(t)$ minus the velocity predicted using the following expression:

$$v_{(o)}(t) \approx v_{(o)}(t-1) + \hat{a}(\pi, t, x_{(o)}(t-1), v_{(o)}(t-1)) \Delta t,$$

where Δt is the time between sensor measurements.

We reduce the number of sensor values by projecting these difference curves down onto a set of basis functions. We chose the same basis functions that we used to encode the policy for the two tasks; we used splines for the dart and a truncated Fourier series for the quadruped. This is the sensor data that we give to the gradient estimator algorithms as described in Section 2.2. We also include a sensor that detects the release time for the dart throwing task.

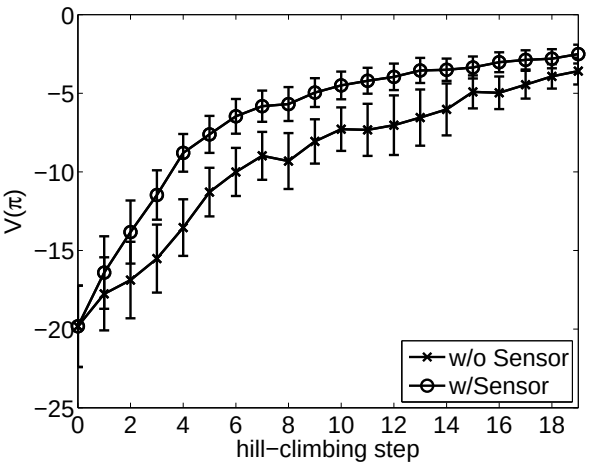


Figure 5: The learning curve performance of two policy search strategies in the dart throwing domain. At each hill-climbing step we drew a single sample from 12 different policies and we averaged over 48 hill-climbing runs.

3.2 LEARNING PERFORMANCE

Figure 5 shows the learning curves for the dart throwing problem. We get a substantial improvement in the learning performance when using an algorithm that incorporates sensor data when compared to an algorithm that ignores the sensor data. At each hill-climbing step we drew a single sample from 12 different policies and we averaged over 48 hill-climbing runs. The policies for each hill-climbing step were drawn from a Gaussian distribution for exploration purposes.

Figure 6 shows the learning curves for the quadruped locomotion problem. We get an improvement in the learning performance when using of an algorithm that incorporates sensor data when compared to an algorithm that ignores the sensor data. At each hill-climbing step we drew a single sample from 30 different policies and we averaged over 48 hill-climbing runs. The policies for each hill-climbing step were drawn from a Gaussian distribution for exploration purposes.

4 DISCUSSION

In this paper we demonstrated how one may incorporate sensor data into the gradient estimation task to improve the performance of policy search. We showed that the level of improvement depends on the amount of information that the sensors provide about

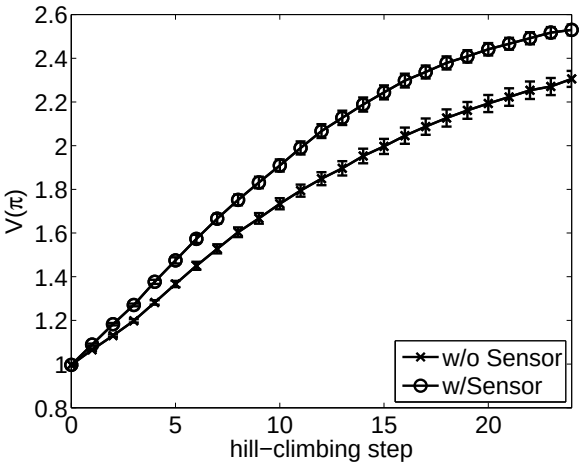


Figure 6: The learning curve performance of two policy search strategies in the quadruped learning domain. At each hill-climbing step we drew a single sample from 30 different policies and we averaged over 48 hill-climbing runs.

the noise-induced perturbations on the score. We also showed that the performance gains depend on the dimensionality of the sensor data. It is important to choose sensor encodings that are independent of the policy parameters to minimize the bias. We also presented a technique to find good sensor encodings for problems in which these assumptions (low-dimensionality and statistical indepedence) do not hold. Finally, we presented learning curves that show improvements in the learning performance for a toy cannon problem, dart throwing task, and quadruped locomotion task.

In this paper the distribution of every random variable was approximated by using a linear Gaussian relationship. Improvements in performance may be realized if we use non-linear mappings. Other improvements may come from incorporating our knowledge of the physics behind each task. For example, in the cannon problem we already know the equations of projectile motion. Thus, given the actual controls, we should be able to accurately predict the score. Even in cases in which we do not know the equations of motion, we often know qualitative information about the motion, such as the fact that increasing the desired velocity causes the cannon ball to fly further (i.e., the distance travelled is monotonically increasing as a function of the desired velocity). One possible approach to incorporating this information is to place a prior on the parameters that reflects these constraints.

References

[1] J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[2] B. Garner and M. Pandy. A kinematic model of the upper limb based on the visible human project (vhp) image dataset. *Computer Methods in Biomechanics and Biomedical Engineering*, 2:107–124, 1999.

[3] Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1507–1514, 2001.

[4] Christopher Harris and Daniel Wolpert. Signal-dependent noise determines motor planning. *Nature*, 394:780–784, 1998.

[5] N. Kohl and P. Stone. Machine learning for fast quadrupedal locomotion. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 2004.

[6] Gregory Lawrence, Noah Cowan, and Stuart Russell. Efficient gradient estimation for motor control learning. In *Proceedings of the Nineteenth International Conference on Uncertainty in Artificial Intelligence*, 2003.

[7] Andrew Y. Ng and Michael Jordan. Pegasus: A policy search method for large MDPs and POMDPs. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 406–415, 2000.

[8] Andrew Y. Ng, H. Jin Kim, Michael Jordan, and Shankar Sastry. Autonomous helicopter flight via reinforcement learning. In *Advances in Neural Information Processing Systems*, 2003.

[9] Leon Peshkin and Christian R. Shelton. Learning from scarce experience. In *Proceedings of the 19th International Conference on Machine Learning*, 2002.

[10] Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Natural actor-critic. In *Proceedings of the 16th European Conference on Machine Learnin*, 2003.

[11] Christian R. Shelton. Policy improvement for POMDPs using normalized importance sampling. In *Proceedings of the Seventeenth International Conference on Uncertainty in Artificial Intelligence*, pages 496–503, 2001.

[12] Emanuel Todorov and Michael I. Jordan. Optimal control as a theory of motor coordination. *Nature Neuroscience*, 2002.

[13] Lex Weaver and Nigel Tao. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 538–545, 2001.

[14] D. Wolpert, Z. Ghahramani, and J. Flanagan. Perspectives and problems in motor learning. *Trends in Cognitive Science*, 5, 2001.

Learning Arithmetic Circuits

Daniel Lowd and **Pedro Domingos**
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350, U.S.A.
{*lowd,pedrod*}*@cs.washington.edu*

Abstract

Graphical models are usually learned without regard to the cost of doing inference with them. As a result, even if a good model is learned, it may perform poorly at prediction, because it requires approximate inference. We propose an alternative: learning models with a score function that directly penalizes the cost of inference. Specifically, we learn arithmetic circuits with a penalty on the number of edges in the circuit (in which the cost of inference is linear). Our algorithm is equivalent to learning a Bayesian network with context-specific independence by greedily splitting conditional distributions, at each step scoring the candidates by compiling the resulting network into an arithmetic circuit, and using its size as the penalty. We show how this can be done efficiently, without compiling a circuit from scratch for each candidate. Experiments on several real-world domains show that our algorithm is able to learn tractable models with very large treewidth, and yields more accurate predictions than a standard context-specific Bayesian network learner, in far less time.

1 INTRODUCTION

Bayesian networks are a powerful language for probabilistic modeling, capable of compactly representing very complex dependences. Unfortunately, the compactness of the representation does not necessarily translate into efficient inference. Networks with relatively few edges per node can still require exponential inference time. As a consequence, approximate inference methods must often be used, but these can yield poor and unreliable results. If the network represents manually encoded expert knowledge, this is perhaps inevitable. But when the network is learned from data, the cost of inference can potentially be greatly reduced, without compromising accuracy, by suitably directing the learning process.

Bayesian networks can be learned using local search to maximize a likelihood or Bayesian score, with operators like edge addition, deletion and reversal (Heckerman et al., 1995). Typically, the number of parameters or edges in the network is penalized to avoid overfitting, but this is only very indirectly related to the cost of inference. Two edge additions that produce the same improvement in likelihood can result in vastly difference inference costs. In this case, it seems reasonable to prefer the edge yielding the lowest inference cost. In this paper, we propose a learning method that accomplishes this, by directly penalizing the cost of inference in the score function.

Our method takes advantage of recent advances in exact inference by compilation to arithmetic circuits (Darwiche, 2003). An arithmetic circuit is a representation of a Bayesian network capable of answering arbitrary marginal and conditional queries, with the property that the cost of inference is linear in the size of the circuit. When context-specific independences are present, arithmetic circuits can be much more compact than the corresponding junction trees. We take advantage of this by learning arithmetic circuits that are equivalent to Bayesian networks with context-specific independence, using likelihood plus a penalty on the circuit size as the score function. Arithmetic circuits can also take advantage of other structural properties such as deterministic dependencies and latent variables; utilizing these in addition to context-specific independence is an important item of future work.

Previous work on learning graphical models with the explicit goal of limiting the complexity of inference falls into two main classes: mixture models with polynomial-time inference (e.g.: Meila and Jordan (2000); Lowd and Domingos (2005)) and graphical models with thin junction trees (e.g.: Srebro (2000); Chechetka and Guestrin (2008)). The former are limited in the range of distributions that they can compactly represent. The latter are computationally viable (at both learning and inference time) only for very low treewidths. Our approach can flexibly and compactly learn a wide variety of models, including models with very large

treewidth, while guaranteeing efficient inference, by taking advantage of the properties of arithmetic circuits.

The prior work most closely related to ours is Jaeger et al.'s (2006). Jaeger et al. define probabilistic decision graphs, a new language related to binary decision diagrams. In contrast, we use standard arithmetic circuits, and our models are equivalent to standard Bayesian networks. Jaeger et al. speculate that learning arithmetic circuits directly from data would be very difficult. In this paper we propose one approach to doing this.

The remainder of our paper is organized as follows. In Sections 2 and 3, we provide background on Bayesian networks and arithmetic circuits, respectively. We describe in detail our algorithm for learning arithmetic circuits in Section 4. Section 5 contains our empirical evaluation on three real-world datasets, and we conclude in Section 6.

2 BAYESIAN NETWORKS

A *Bayesian network* encodes the joint probability distribution of a set of n variables, $\{X_1, \ldots, X_n\}$, as a directed acyclic graph and a set of conditional probability distributions (CPDs) (Pearl, 1988). Each node corresponds to a variable, and the CPD associated with it gives the probability of each state of the variable given every possible combination of states of its parents. The set of parents of X_i, denoted Π_i, is the set of nodes with an arc to X_i in the graph. The structure of the network encodes the assertion that each node is conditionally independent of its non-descendants given its parents. The joint distribution of the variables is thus given by $P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\Pi_i)$.

For discrete domains, the simplest form of CPD is a conditional probability table. When the structure of the network is known, learning reduces to estimating CPD parameters. When the structure is unknown, it can be learned by starting with an empty or prior network and greedily adding, deleting and reversing arcs to optimize some score function (Heckerman et al., 1995). The score function is usually log-likelihood plus a complexity penalty or a Bayesian score (product of prior and marginal likelihood).

The goal of inference in Bayesian networks is to answer arbitrary marginal and conditional queries (i.e., to compute the marginal distribution of a set of query variables, possibly conditioned on the values of a set of evidence variables). One common method is to construct a *junction tree* from the Bayesian network and pass messages from the leaves of this tree to the root and back. A junction tree is constructed by connecting parents of the same variable, removing arrows, and triangulating the resulting undirected graph (i.e., ensuring that all cycles of length four or more have a chord). Each node in the junction tree corresponds to a *clique* (maximal completely connected subset of variables) in the triangulated graph. Ordering cliques by the highest-ranked variable they contain, each clique is connected to a predecessor sharing the highest number of variables with it. The intersection of the variables in two adjacent cliques is called the *separator* of the two cliques. A junction tree satisfies two important properties: each variable in the Bayesian network appears in some clique with all of its parents; and if a variable appears in two cliques, it appears in all the cliques on the path between them (the *running intersection property*). The *treewidth* of a junction tree is one less than the maximum clique size. The complexity of inference is exponential in the treewidth. Finding the minimum-treewidth junction tree is NP-hard (Arnborg et al., 1987). Inference in Bayesian networks is #P-complete (Roth, 1996).

Because exact inference is intractable, approximate methods are often used, of which the most popular is *Gibbs sampling*, a form of Markov chain Monte Carlo (Gilks et al., 1996). A Gibbs sampler proceeds by sampling each non-evidence variable in turn conditioned on its Markov blanket (parents, children and parents of children). The distribution of the query variables is then approximated by computing, for each possible state of the variables, the fraction of samples in which it occurs. Gibbs sampling can be very slow to converge, and many MCMC variations have been developed, but choosing and tuning one for a given application remains a difficult, labor-intensive task. Diagnosing convergence is also difficult.

2.1 LOCAL STRUCTURE

Table CPDs require exponential space in the number of parents of the variable. A more scalable approach is to use *decision trees* as CPDs, taking advantage of context-specific independencies (i.e., a child variable is independent of some of its parents given some values of the others) (Boutilier et al., 1996; Friedman & Goldszmidt, 1996; Chickering et al., 1997). The algorithm we present in this paper learns arithmetic circuits that are equivalent to this type of Bayesian network.

In a decision tree CPD for variable X_i, each interior node is labeled with one of the parent variables, and each of its outgoing edges is labeled with a value of that variable.[1] Each leaf node is a multinomial representing the marginal distribution of X_i conditioned on the parent variable values specified by its ancestor nodes and edges in the tree.

The following two definitions will be useful in describing our algorithm.

[1] In general, each outgoing edge can be labeled with any subset of the variable's values, as long as the sets of labels assigned to all child edges include every variable value and are disjoint with each other. For simplicity, we limit our discussion to the case in which each edge has a single label, which Chickering et al. (1997) refer to as a *complete split*. For Boolean variables, as in our experiments, all types of splits are equivalent.

Definition 1. *For leaf node D and k-valued variable X_j, the* split $S(D, X_j)$ *replaces D with k new leaves, each conditioned on a particular value of X_j in addition to the parent values on the path to D.*

Definition 2. *Let D be a leaf from the tree CPD for X_i. Split $S(D, X_j)$ is* valid *iff X_j is not a descendant of X_i in the Bayesian network and no decision tree ancestor of D is labeled with X_j.*

The first definition describes a structural update to the Bayesian network; the second one gives the conditions necessary for that update to be consistent and meaningful.

A Bayesian network can now be learned by greedily applying the best valid splits according to some criterion, such as the likelihood of the data penalized by the number of parameters. This is one version of Chickering et al.'s algorithm (1997). A number of other methods have also been proposed, such as merging leaves to obtain decision graphs (Chickering et al., 1997) or searching through Bayesian network structures and inducing decision trees conditioned on the global structure (Friedman & Goldszmidt, 1996).

3 ARITHMETIC CIRCUITS

The probability distribution represented by a Bayesian network can be equivalently represented by a multilinear function known as the *network polynomial* (Darwiche, 2003):

$$\begin{aligned} & P(X_1 = x_1, \ldots, X_n = x_n) \\ &= \sum_{\mathbf{X}} \prod_{i=1}^{n} I(X_i = x_i) P(X_i = x_i | \Pi_i = \pi_i) \end{aligned}$$

where the sum ranges over all possible instantiations of the variables, $I()$ is the indicator function (1 if the argument is true, 0 otherwise), and the $P(X_i|\Pi_i)$ are the parameters of the Bayesian network. The probability of any partial instantiation of the variables can now be computed simply by setting to 1 all indicators consistent with the instantiation, and to 0 all others. This allows arbitrary marginal and conditional queries to be answered in time linear in the size of the polynomial.

Unfortunately, the size of the network polynomial is exponential in the number of variables, but it can be more compactly represented using an *arithmetic circuit*. An arithmetic circuit is a rooted, directed acyclic graph whose leaves are numeric constants or variables, and whose interior nodes are addition and multiplication operations. The value of the function for an input tuple is computed by setting the variable leaves to the corresponding values and computing the value of each node from the values of its children, starting at the leaves. In the case of the network polynomial, the leaves are the indicators and network parameters. The arithmetic circuit avoids the redundancy present in the network polynomial, and can be exponentially more compact.

Every junction tree has a corresponding arithmetic circuit, with an addition node for every instantiation of a separator, a multiplication node for every instantiation of a clique, and an addition node as the root. Thus one way to compile a Bayesian network into an arithmetic circuit is via a junction tree. However, when the network contains context-specific independences, a much more compact circuit can be obtained. Darwiche (2003) describes one way to do this, by encoding the network into a special logical form, factoring the logical form, and extracting the corresponding arithmetic circuit.

4 LEARNING ARITHMETIC CIRCUITS

4.1 SCORING AND SEARCHING

Instead of learning a Bayesian network and then compiling it into a circuit, we induce an arithmetic circuit directly from data using a score function that penalizes circuits with more edges. The score of an arithmetic circuit C on an i.i.d. training sample T is

$$\text{score}(C, T) = \log P(T|C) - k_e n_e(C) - k_p n_p(C)$$

where the first term is the log-likelihood of the training data, $P(T|C) = \prod_{X \in T} P(X|C)$, $k_e \geq 0$ is the per-edge penalty, $n_e(C)$ is the number of edges in the circuit, $k_p \geq 0$ is the per-parameter penalty, and $n_p(C)$ is the number of parameters in the circuit. The last two allow us to easily combine our inference-cost penalty with a more traditional one based on model complexity.

We use this formulation for simplicity; our algorithm would work equally well with a Bayesian Dirichlet score (Heckerman et al., 1995), with a prior of the form $\exp(-k_e n_e(C) - k_p n_p(C))$, since the computation of the marginal likelihood would be the same as in standard Bayesian network learning. Aside from its practical utility, a prior penalizing inference cost is meaningful if we believe the inference task being modeled can be carried out quickly, for example because humans do it. Either way, the main difficulty is that the penalty (or prior) is no longer node-decomposable, and repeatedly computing it might be very expensive. Reducing this cost is one of the key technical issues addressed in this paper.

Arithmetic circuits can be learned in the same way as Bayesian networks with local structure, by starting with an empty network and greedily applying the best splits, except that candidate structures are scored by compiling them into arithmetic circuits. However, compiling an arithmetic circuit can be computationally costly, and doing so for every candidate structure would be prohibitive. A better approach is to incrementally compile the circuit as splits are applied. Table 1 shows pseudo-code for this algorithm.

The algorithm begins by constructing the initial arithmetic

Table 1: Greedy algorithm for learning arithmetic circuits.

function LearnAC(T)
initialize circuit C as product of marginals
loop
  $C_{best} \leftarrow C$
  for each valid split $S(D, V)$ **do**
    $C' \leftarrow$ SplitAC$(C, S(D, V))$
    if score$(C', T) >$ score(C_{best}, T) **then**
      $C_{best} \leftarrow C'$
    end if
  end for
  if score$(C_{best}, T) >$ score(C, T) **then**
    $C \leftarrow C_{best}$
  else
    return C
  end if
end loop

circuit C as a product of marginal distributions:

$$C = \prod_i \sum_j I(X_i = x_{ij}) P(X_i = x_{ij})$$

This initial circuit is equivalent to a Bayesian network with no edges. In each iteration, the algorithm greedily chooses and applies the best valid split, where split validity is defined according to the equivalent Bayesian network. Each split is scored by applying it to the current circuit and counting the edges and parameters.[2] Learning ends when the algorithm reaches a local optimum, where no valid split improves the score.

4.2 SPLITTING DISTRIBUTIONS

The key subroutine is SplitAC, which updates an arithmetic circuit without recompiling it from scratch. Given an arithmetic circuit C that is equivalent to a Bayesian network B and a valid split $S(D, V)$, SplitAC returns a modified circuit C' that is equivalent to B after applying split $S(D, V)$. We will use the following notation to refer to distributions, parameter nodes, and indicator nodes:

d_j**:** Parameter node corresponding to the jth probability in the multinomial distribution D.

D_i**:** Leaf distribution resulting from split $S(D, V)$ that replaces D when $V = i$.

d_{ij}**:** Parameter node corresponding to the jth probability in D_i.

v_i**:** Indicator node $I(V = i)$.

[2]All model parameters are MAP estimates, using a Dirichlet prior with all hyperparameters $\alpha_{ijk} = 1$, where k ranges over the leaves of the decision tree for variable X_i.

Table 2: Subroutine that updates an arithmetic circuit C by splitting distribution D on variable V.

function SplitAC($C, S(D, V)$)
let M be the set of mutual ancestors of D and V
let N be the set of nodes between M and V or D
for $i \in$ Domain(V) **do**
  create new parameter nodes d_{ij}
  $N_i \leftarrow$ copy of all nodes in N
  for each $n \in N$ **do**
    let n_i be the copy of n in N_i
    for each child c of n **do**
      if $c = v_i$ or c is inconsistent with v_i **then**
        skip
      else if c is some parameter node d_j **then**
        insert edge from n_i to d_{ij}
      else if $c \in N$ **then**
        let c_i be the copy of c in N_i
        insert edge from n_i to c_i
      else
        insert edge from n_i to c
      end if
    end for
  end for
end for
for $m \in M$ **do**
  let n_V be the child of m that is a V-ancestor
  let n_D be the child of m that is a D-ancestor
  for $i \in$ Domain(V) **do**
    let n'_V be the copy of n_V in N_i
    let n'_D be the copy of n_D in N_i
    create $n_{\times_i} := v_i \times n'_V \times n'_D$
  end for
  create $n_+ := \sum_i n_{\times_i}$
  replace m's children n_V and n_D with n_+
end for
delete unreachable nodes, including all d_j

Table 2 contains pseudo-code for the splitting algorithm. It might at first appear that to split D on V it suffices to replace references to each d_j with a sum of products, $\sum_i d_{ij} v_i$. However, the resulting circuit would then be correct only when V is fixed to a particular value, and summing out V would produce inconsistent results. Intuitively, the circuit must maintain the running intersection property of the corresponding junction tree, so that no variable can take on different values in different subcircuits. SplitAC maintains a consistent probability distribution by preserving three properties, analogous to those defined by Darwiche (2002) for logical circuits.

Definition 3. *For an arithmetic circuit, C:*

C is smooth *if, for each addition node, all children are ancestors of indicator nodes for the same variables and pa-*

rameter nodes from distributions of the same variables.

C *is* decomposable *if, for each multiplication node, no two children are ancestors of indicator nodes for the same variable or parameter nodes from distributions of the same variable.*

C *is* deterministic *if, for each addition node, there is a variable V such that each child is the ancestor of some non-empty set of indicator nodes for V, and their sets are disjoint.*

The network polynomial for a Bayesian network contains one term for each configuration of its variables; each term includes exactly one indicator variable and one conditional probability parameter per variable. Intuitively, if C is not smooth, then some terms in the polynomial it computes may not have an indicator variable and a conditional probability parameter for every variable. If C is not decomposable, then some terms in the polynomial may have more than one indicator variable or conditional probability parameter for some variable. If C is not deterministic, then there may be multiple terms for the same set of indicator variables.

Definition 4. *We define three special types of node in the circuit as follows:*

A D-ancestor *is any leaf d_j corresponding to a parameter of D, or any parent of a D-ancestor.*

A V-ancestor *is any leaf v_i corresponding to an indicator of V, or any parent of a V-ancestor.*

A mutual ancestor (MA) of D and V *is a node that is both a D-ancestor and a V-ancestor, and has no child that is both a D-ancestor and a V-ancestor.*

Note that every MA must be a multiplication node, or the circuit would not be smooth. Furthermore, from decomposability, each MA must have exactly one D-ancestor child, n_D, and one V-ancestor child, n_V. Naively replacing d_j with $\sum_i d_{ij} v_i$ would cause both n_V and n_D to be ancestors of v_i, violating decomposability.

To avoid this, SplitAC duplicates the subcircuits between the MAs and the parameter nodes d_j, and between the MAs and the indicator nodes v_i, "conditioning" each copy on a different value of V. Each n_V and n_D are replaced by a new addition node, n_+, that sums over products of v_i and copies of n_V and n_D conditioned on v_i. This duplication of subcircuits is the reason different splits can have widely different edge costs. We now describe the details of which nodes are duplicated and how they are connected.

Let N be the set of all D-ancestors and V-ancestors that are also descendants of a mutual ancestor. These are all the nodes "in between" D and V that must agree on the value of V. For each value i in the domain of V, SplitAC creates a copy N_i of the nodes in N.

Let $n_i \in N_i$ be the copy of node $n \in N$. SplitAC inserts edges from n_i to its children as follows. If n has a child $c \in N$, then it inserts an edge from n_i to the corresponding copy c_i. If n has a child $c \notin N$, then it inserts an edge from n_i to c. This minimizes node duplication by linking to existing nodes or copies whenever possible.

A few additional changes are required for N_i to properly depend on v_i. If $n_i \in N_i$ has some parameter node d_j as a child, SplitAC replaces it with d_{ij}. This is how the new leaf distributions, conditioned on V, are integrated into the circuit. Secondly, if n_i has v_i as a child, it should be omitted: every node in N_i will depend on v_i, so this is redundant. Finally, if n_i has a child that is an ancestor of some v_j but not of v_i, then that child is inconsistent with conditioning on v_i and must be removed.

Finally, SplitAC connects each mutual ancestor, m, to a sum over these copies. SplitAC removes the D-ancestor, n_D, and the V-ancestor, n_V, as children of m and replaces them with an addition node with one child for each value of V. The ith child of the addition node is a product of v_i, the copy of n_D from N_i, and the copy of n_V from N_i. (If m was an ancestor of only certain values of V, the addition node sums only over those values.)

Intuitively, the resulting circuit represents the correct probability distribution because D has been replaced with the split distributions D_i, each conditioned on v_i, and because the circuit satisfies the running intersection property, since all nodes between V and D now depend on V.

Theorem 1. *After each iteration of LearnAC, C computes the network polynomial of a Bayesian network constructed by starting with an empty network and applying the same splits that were applied to C up to that iteration.*

A proof sketch is in the appendix; a complete proof can be found in (Lowd & Domingos, 2008).

4.3 OPTIMIZATIONS

We now discuss optimizations necessary to make this algorithm practical for real-world datasets with many variables.

Consider once again the high-level overview in Table 1. Scoring every possible circuit in every iteration would be very expensive. Choosing the split that leads to the best scoring circuit is equivalent to choosing the split that leads to the greatest increase in score, so we can store changes in score instead. The improvement in log-likelihood is not affected by other splits, and so this only needs to be computed once for each potential split. Unfortunately, the number of edges that a split adds to the circuit can increase or decrease due to other splits. For convenience, we will refer to the number of edges added by the application of a split as its *edge cost*.

As a simple example, consider a chain-structured junction

tree of 5 variables: AB-BC-CD-DE-EF. If we add an arc from A to F, then A is added to every other cluster: AB-ABC-ACD-ADE-AEF. However, this also reduces the cost of adding an arc from A to E, since the two variables now appear together in a cluster. As a second example, suppose that we instead added an arc from B to F: AB-BC-BCD-BDE-BEF. Now the cost of adding an arc from A to F is greatly increased, since adding a variable to a larger cluster costs more edges than adding a variable to a smaller cluster.

Evaluating the edge cost of every potential split in every iteration is expensive. The number of potential splits is linear in the number of splits that have been performed so far, leading to a time complexity that is at least quadratic in the total number of splits. Further, computing the edge cost for a single candidate may be linear in the size of the current circuit. With a non-zero edge cost, circuit size tends to be linear in the number of iterations, leading to an $O(n^3)$ algorithm. While this is still polynomial, it makes learning models with thousands of splits intractable in practice.

Fortunately, most splits only change a fraction of edge costs. Determining exactly which costs need to be updated is difficult, but we can rule out many splits whose costs do not need to be updated using the following conservative rule. Applying one split may change the edge cost of another split $S(D, V)$ if the applied split changes a node that is an ancestor of D and not V, or of V and not D. This covers all nodes that lie between D or V and their mutual ancestors, and thus all nodes that are copied by the splitting procedure. An applied split changes a node when it copies that node or reduces the number of children it has. In practice, this single heuristic lets us avoid recomputing over 95% of the edge costs.

As an alternative to this optimization, we have found a heuristic that leads to even larger speed-ups, but at the cost of no longer being perfectly greedy. We noticed that when edge costs changed, they rarely decreased. If a split's last computed edge cost was always a valid lower bound on the true value, then we could ignore any split whose total estimated score was worse than the best split found so far in this iteration. This assumption is often not valid in practice, but it lets us learn models that are nearly as effective in an order of magnitude less time.

Two other optimizations combine well with either of the above to offer further gains. First, we can reduce the number of computations by placing potential splits in order of decreasing likelihood gain, so that we consider the splits with the highest possible scores first. Since the likelihood gain is an upper bound on the score gain, once the score of the best split found so far is greater than the next likelihood gain, this split is guaranteed to be the highest-scoring one overall.

Second, we can exit the edge calculation procedure once we know that the edge cost is sufficient to make the overall score negative. It is also possible to exit once we know that the score of the current split will be worse than the best split so far, but this interferes with the other optimizations. If we only compute an upper bound on the score, we will often have to recompute the edge cost when the next iteration requires a slightly lower upper bound.

Table 3: Summary of experimental datasets.

Domain	Vars.	Train Exs.	Test Exs.	Density
KDD Cup	65	199,999	34,955	0.0079
MSWeb	294	32,711	5,000	0.0102
EachMovie	500	6,117	591	0.0581

5 EXPERIMENTS

5.1 DATASETS

We evaluated our methods on three widely used real-world datasets. The KDD Cup 2000 clickstream prediction dataset (Kohavi et al., 2000) consists of web session data taken from an online retailer. Using the subset of Hulten and Domingos (2002), each example consists of 65 Boolean variables, corresponding to whether or not a particular session visited a web page matching a certain category. Anonymous MSWeb is visit data for 294 areas (Vroots) of the Microsoft web site, collected during one week in February 1998. It can be found in the UCI machine learning repository (Blake & Merz, 2000). EachMovie[3] is a collaborative filtering dataset in which users rate movies they have seen. We took a 10% sample of the original dataset, focused on the 500 most-rated movies, and reduced each variable to "rated" or "not rated". For KDD Cup and MSWeb, we used the training and test partitions provided with the datasets. For EachMovie, we randomly selected 10% of the data for the test set and used the remainder for training.

Basic statistics for each dataset are shown in Table 3. Density refers to the fraction of non-zero entries across all examples and all variables.

5.2 LEARNING

For each dataset, we randomly split the training data into tuning and validation sets, corresponding to 90% and 10% of the training data, respectively. All parameters were tuned by training models on the tuning data and selecting the parameter sets that led to the highest log likelihood of the validation set. Finally, models were retrained using the full training set. All experiments were run on CPUs with 4 GB of RAM running at 2.8 GHz.

[3]Provided by Compaq at http://research.compaq.com/SRC/-eachmovie/; no longer available for download, as of October 2004.

We used two versions of the algorithm for learning arithmetic circuits from Section 4: AC-Greedy, which guarantees that we pick the best split in each iteration, and AC-Quick, which uses a heuristic to avoid recomputing edge costs but may sometimes choose worse splits. We varied the per-edge cost k_e from 1.0 to 0.01. Not surprisingly, our models were most accurate on the validation set with low per-edge costs (0.01 or 0.02). We also tuned the per-parameter cost k_p. For KDD Cup, the best cost was 0.0; for MSWeb and EachMovie, the best costs were 1.0 for greedy ACs and 0.5 for quick ACs.

We used the WinMine Toolkit (Chickering, 2002) as a baseline. WinMine implements the algorithm for learning Bayesian networks with local structure described in Section 2 (Chickering et al., 1997), and has a number of other state-of-the-art features. We tuned WinMine's multiplicative per-parameter penalty κ; the best values were: 1 (no penalty) for KDD Cup, 0.1 for MSWeb, and 0.01 for EachMovie. We looked into using thin junction trees as a second baseline, but they do not scale to datasets of these dimensions.

A summary of the learned models appears in Table 4. For each dataset, we report the log-likelihood per example on the test data, the number of edges in the arithmetic circuit, the number of leaves across all decision trees, the average and maximum number of parents across all variables, the treewidth (estimated using a min-fill heuristic), the number of edges generated by compiling the Bayesian network using c2d[4], and the training time. On each model for which c2d ran out of memory, we obtained a lower bound by compiling a model with fewer splits, obtained by halting the learning process early. We varied the number of splits until we found the most complex sub-model that could still be compiled, within 10 splits. For WinMine, the chosen sub-models had less than one quarter of the original splits.

The test-set log-likelihoods of the AC learners and WinMine are very similar, with WinMine having a slight edge. This is not surprising, given that WinMine is free to choose expensive splits. Perhaps more remarkable is that this freedom translates to very little improvement in likelihood. The difference in accuracy between quick and greedy ACs is negligible except in the case of EachMovie, where the greedy AC is actually less accurate because it did not converge in the allowed time (72h).

Not surprisingly, WinMine is much faster than the AC learners. It is worth noting that the cost of learning is only incurred once, while the cost of inference is incurred many times. Also, the AC learner directly outputs an arithmetic circuit, while WinMine's Bayesian network would still have to be compiled into one, which can be very time-consuming. Finally, the quick heuristic offers up to an order-of-magnitude speedup with similar accuracy; additional heuristics might offer additional improvements.

Table 4: Summary of Learned Models

KDD Cup	AC-Greedy	AC-Quick	WinMine
Log-likelih.	-2.16	-2.16	-2.16
Edges	382K	365K	
Leaves	4574	4463	2267
Avg. parents	13.2	13.0	16.3
Max. parents	37	36	35
Treewidth	38	38	53
c2d edges	>18.2M	3664k	>39.5M
Time	50h	3h	3m

MSWeb	AC-Greedy	AC-Quick	WinMine
Log-likelih.	-9.85	-9.85	-9.69
Edges	204K	256K	
Leaves	1353	1870	1710
Avg. parents	2.5	3.1	5.2
Max. parents	114	127	94
Treewidth	114	127	118
c2d edges	>23.5M	>44.6M	>63.5M
Time	8h	3h	2m

EachMovie	AC-Greedy	AC-Quick	WinMine
Log-likelih.	-55.7	-54.9	-53.7
Edges	155K	372K	
Leaves	4070	6521	4830
Avg. parents	5.0	6.5	8.0
Max. parents	13	17	27
Treewidth	35	54	281
c2d edges	207k	855k	>27.3M
Time	>72h[5]	22h	3m

5.3 INFERENCE

For each dataset, we used the test data to generate queries with varied numbers of randomly selected query and evidence variables. Each query asked the probability of the configuration of the query variables in the test example conditioned on the configuration of the evidence variables in the same test example.

We estimate inference accuracy as the mean log probability of the test examples's configuration across all test examples. This is an approximation (up to an additive constant) of the Kullback-Leibler divergence between the inferred distribution and the true one, estimated using the test samples. For KDD Cup and MSWeb, we generated queries from 1000 test examples; for EachMovie, we gen-

[4]Available at http://reasoning.cs.ucla.edu/c2d/. We also tried using the ACE package, but it does not support decision tree CPDs and, for our models, tabular CPDs would be prohibitively large.

[5]AC-Greedy did not finish running in the maximum allowed time of 72h. As a result, it has fewer edges and lower log-likelihood than AC-Quick.

Table 5: Average inference time per query.

Algorithm	KDD Cup	MSWeb	EachMovie
AC-Greedy	194ms	91ms	62ms
AC-Quick	198ms	115ms	162ms
Gibbs-Fast	1.46s	1.89s	7.22s
Gibbs-Medium	11.3s	15.6s	42.5s
Gibbs-Slow	106s	154s	452s
Gibbs-VerySlow	1124s	1556s	3912s

erated queries from all 593 test examples.

For the arithmetic circuits, we used exact inference. For the Bayesian networks learned using WinMine, we used Gibbs sampling. We initialized the sampler to a random state, ran it for a burn-in period, and then collected samples to estimate the probability of the queried marginal or conditional event. All estimates were smoothed by uniformly distributing a count of 1 across all states of the query variables. Since convergence is difficult to diagnose and may take prohibitively long, we ran Gibbs sampling in four scenarios: fast (one chain, 100 burn-in iterations, 1000 sampling iterations); medium (ten chains, 100 burn-in iterations, 1000 sampling iterations); slow (ten chains, 1000 burn-in iterations, 10,000 sampling iterations); and very slow (ten chains, 10,000 burn-in iterations, 100,000 sampling iterations).

Figure 1 shows the relative accuracy of the different methods on each dataset. Per-variable query log-likelihood is on the y axis. In the graphs on the left, each query included 30% of the variables in the domain, conditioned on 0% to 50% of the domain variables as evidence. In the graphs on the right, the number of query variables varies from 10% to 50%, conditioned on 30% of the variables in the domain as evidence. Inference times (averaged over all queries) are listed in Table 5. Note that AC inference times are in milliseconds, while Gibbs inference times are in seconds.

The ACs were roughly one order of magnitude faster than the fastest runs of Gibbs sampling, and four orders of magnitude faster than the slowest. Except when the number of query variables is very small, the ACs also easily dominate even the slowest runs of Gibbs sampling on accuracy. Because of the approximate inference, the slightly higher test-set log-likelihood of WinMine's models does not translate into higher accuracy in answering queries. Presumably, given enough time Gibbs sampling will eventually catch up with the ACs in accuracy, but by then it will be many orders of magnitude slower. Further, Gibbs sampling (like other approximate inference methods) requires tuning for best results, and we can never be sure that it has converged. In contrast, the AC inference is reliable, the time it takes is predetermined, and the time is short enough for online or interactive use.

6 CONCLUSION

In the past, work on learning and inference in graphical models has been largely separate. This has had the somewhat paradoxical result that much effort is often expended to learn accurate models, only to result in less accurate predictions when approximate inference becomes necessary. Our work seeks to ameliorate this by more closely integrating learning and inference. In particular, we presented an algorithm for learning arithmetic circuits by maximizing likelihood with a penalty on circuit size. This ensures efficient inference while still providing great modeling flexibility. In experiments on real-world domains, our algorithm outperformed standard Bayesian network learning on both accuracy of query answers and speed of inference.

Directions for future work include: investigating other algorithms for learning arithmetic circuits; extending our approach to handle learning with missing data and hidden variables; applying it to Markov networks, continuous domains, and relational representations; etc.

Acknowledgements

The authors wish to thank Mark Chavira, Adnan Darwiche, and Knot Pipatsrisawat for help applying c2d to our Bayesian networks. This research was partly funded by a Microsoft Research fellowship awarded to the first author, DARPA contracts NBCH-D030010/02-000225, FA8750-07-D-0185, and HR0011-07-C-0060, DARPA grant FA8750-05-2-0283, NSF grant IIS-0534881, and ONR grant N-00014-05-1-0313. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, NSF, ONR, or the United States Government.

References

Arnborg, S., Corneil, D. W., & Proskurowski, A. (1987). Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic & Discrete Methods*, *8*, 277–284.

Blake, C., & Merz, C. J. (2000). *UCI repository of machine learning databases*. Dept. ICS, UC Irvine, CA. http://-www.ics.uci.edu/∼mlearn/MLRepository.html.

Boutilier, C., Friedman, N., Goldszmidt, M., & Koller, D. (1996). Context-specific independence in Bayesian networks. *Proc. UAI-96* (pp. 115–123).

Chechetka, A., & Guestrin, C. (2008). Efficient principled learning of thin junction trees. In *NIPS 20*.

Chickering, D., Heckerman, D., & Meek, C. (1997). A Bayesian approach to learning Bayesian networks with local structure. *Proc. UAI-97* (pp. 80–89).

Chickering, D. M. (2002). *The WinMine toolkit* (Tech. Rept. MSR-TR-2002-103). Microsoft, Redmond, WA.

Darwiche, A. (2002). A logical approach to factoring belief networks. *Proc. KR-02* (pp. 409–420).

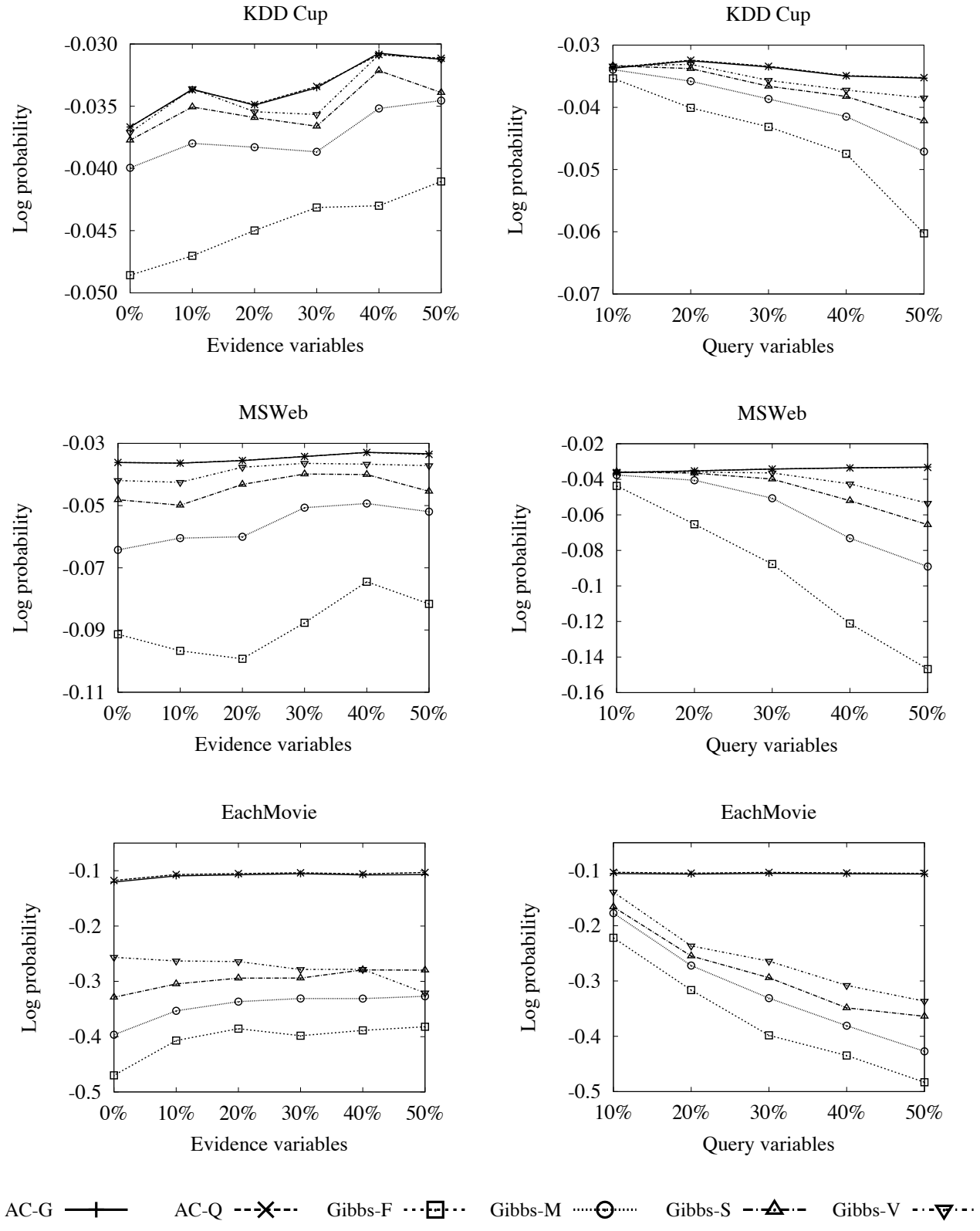


Figure 1: Conditional log probability per query variable, per query. In the legend, AC-G refers to AC-Greedy and AC-Q refers to AC-Quick. Gibbs-F, Gibbs-M, Gibbs-S and Gibbs-V refer to the fast, medium, slow, and very slow Gibbs sampling scenarios, respectively.

Darwiche, A. (2003). A differential approach to inference in Bayesian networks. *J. ACM*, *50*, 280–305.

Friedman, N., & Goldszmidt, M. (1996). Learning Bayesian networks with local structure. *Proc. UAI-96* (pp. 252–262).

Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (Eds.). (1996). *Markov chain Monte Carlo in practice*. Chapman and Hall.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks. *Mach. Learn.*, *20*, 197–243.

Hulten, G., & Domingos, P. (2002). Mining complex models from arbitrarily large databases in constant time. *Proc. KDD-02* (pp. 525–531).

Jaeger, M., Nielsen, J., & Silander, T. (2006). Learning probabilistic decision graphs. *Intl. J. Approx. Reasoning*, *42*, 84–100.

Kohavi, R., Brodley, C., Frasca, B., Mason, L., & Zheng, Z. (2000). KDD-Cup 2000 organizers' report: Peeling the onion. *SIGKDD Explorations*, *2*, 86–98.

Lowd, D., & Domingos, P. (2005). Naive Bayes models for probability estimation. *Proc. ICML-05* (pp. 529–536).

Lowd, D., & Domingos, P. (2008). *Learning arithmetic circuits* (Tech. Rept.). Dept. CSE, Univ. Washington, Seattle, WA. http://www.cs.washington.edu/homes/∼lowd/-lactr.pdf

Meila, M., & Jordan, M. (2000). Learning with mixtures of trees. *J. Mach. Learn. Research*, *1*, 1–48.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann.

Roth, D. (1996). On the hardness of approximate reasoning. *Artif. Intel.*, *82*, 273–302.

Srebro, N. (2000). Maximum likelihood Markov networks: An algorithmic approach. Master's thesis, MIT, Cambridge, MA.

APPENDIX: PROOF SKETCH FOR THEOREM 1

Lemma 2. *At every iteration of LearnAC, C is smooth, decomposable, and deterministic.*

This can be proved by induction on the calls to SplitAC in each iteration. It is easy to verify that the initial circuit is smooth, decomposable, and deterministic. Verifying that these properties are preserved by each call to SplitAC involves a second induction over the structure of the circuit, working up from the leaf nodes. The proof can be found in Lowd and Domingos (2008).

Proof Sketch for Theorem 1. (By induction over the number of splits performed.) The initial circuit is a product of marginal distributions, equivalent to a Bayesian network with no arcs, so the base case is satisfied.

Assuming the circuit C was equivalent to a Bayesian network B after the last iteration of LearnAC, we must demonstrate that, after applying split $S(D, V)$, the resulting circuit C' is equivalent to B with the split $S(D, V)$.

For an arithmetic circuit, C, we can construct the *logical image* of C by replacing addition with disjunction and multiplication with conjunction. In order to make the different values of each variable mutually exclusive, we replace indicator nodes v_i with conjunctions of v_i and the negation of every other v_j for $j \neq i$. We apply an analogous transformation to the conditional probability parameters for each variable.

It is straightforward to show that if C is a smooth, decomposable, and deterministic AC, then its logical image satisfies the equivalent properties of a logical circuit, as defined by Darwiche (2002).

Let L be the logical image of C and L' be the logical image of C'. From Lemma 2 and the discussion of logical images, we know that C, C', L, and L' are all smooth, deterministic, and decomposable.

It can be shown inductively that the models of L are the terms of the network polynomial for B. It can also be shown that exactly one indicator variable v_i is true for each variable V in every model of L and L'. This means that each logical circuit can be expressed as a disjunction over the values of V: $L = \vee_i(v_i \wedge L)$, $L' = \vee_i(v_i \wedge L')$. In every model of $(v_i \wedge L)$, a node that is an ancestor of v_j and not of v_i is guaranteed to be false. (This can be shown using smoothness.) We can therefore remove links in $(v_i \wedge L)$ to any such node from nodes in between MAs and V without affecting the truth value of the logical circuit.

We can also simplify $(v_i \wedge L')$. For any MA in L', the new addition node (disjunction in L') can be replaced with its ith child since all other children are known to be false. The ith child is a conjunction of v_i (already assumed to be true) and a copy of two children of the MA conditioned on v_i. Since conjunction is associative, we can simplify the MA by linking it directly to the children of this conjunction rather than to the conjunction.

Having performed these simplifications, we can see that: $(v_i \wedge L)$ is logically equivalent to $(v_i \wedge L')$, except that parameters d_j have been replaced with d_{ij} in $(v_i \wedge L')$. Taking the disjunction over all v_i, we can conclude that the models of L are identical to those of L', except that whenever d_j and v_i are true in a model of L, d_{ij} and v_i are true in the corresponding model of L'. This is sufficient to demonstrate that the models of L' are the terms of the network polynomial for B after applying split $S(D, V)$.

Since L' is smooth, deterministic, and decomposable, by Theorem 1 from Darwiche (2002), C' computes the network polynomial of B with the split $S(D, V)$. □

Estimation and Clustering with Infinite Rankings

Marina Meilă and **Le Bao**
Department of Statistics
University of Washington
Seattle, WA 98195-4322
{mmp,lebao}@stat.washington.edu

Abstract

This paper presents a natural extension of stagewise ranking to the the case of infinitely many items. We introduce the infinite generalized Mallows model (IGM), describe its properties and give procedures to estimate it from data. For estimation of multimodal distributions we introduce the *Exponential-Blurring-Mean-Shift* nonparametric clustering algorithm. The experiments highlight the properties of the new model and demonstrate that infinite models can be simple, elegant and practical.

1 Introduction

The stagewise ranking model of [6], also known as *generalized Mallows (GM)*, has been recognized as particularly appropriate for modeling the human process of ranking. This model assigns a permutation π over n items a probability that decays exponentially with its distance to a *central permutation* σ. Here we study this class of models in the limit $n \to \infty$, with the assumption that out of the infinitely many items ordered, one only observes those occupying the first t ranks.

Ordering an infinite number of items is common in retrieval tasks: search engines, programs that match a face, or a fingerprint, or a biological sequence against a database, all output the first t items in a ordering over virtually infinitely many objects. We shall call this output a *top-t* ordering. Unlike machines, people can only reliably rank a small number of items. The GM model has been successfully used to model human ranking decisions. We can view the difference between the standard GM model and the *infinite GM model* that we introduce here as the difference between an election where each voter returns an ordering of a small number of preselected candidates (nominees) and a "grassroots" election process, where everyone can nominate and order their own favourites from a virtually unlimited population. For instance, the difference between "Order the following issues by how much you care about them" vs. "List in order the issues that you care most about" illustrates the difference between the standard and the infinite GM models. By these examples, we argue that the infinite GM corresponds to realistic scenarios.

After defining the infinite GM model, we show that it has sufficient statistics and give algorithms for estimating its parameters from data in the Maximum Likelihood (ML) framework. To be noted that our model will have an infinite number of parameters, of which only a finite number will be constrained by the data from any finite sample. The existence of sufficient statistics also enables us to construct and characterize the conjugate prior for this class of models.

Then, we consider the clustering of top-t ranking data. and introduce an adapted version of the well known Gaussian Blurring Mean-Shift algorithm [2] (GBMS) that we call Exponential Blurring Mean Shift (EBMS).

2 Finite and infinite permutations and their top-t orderings

We consider permutations σ over the set of positive natural numbers $\mathbb{N}^* = \{1, 2, \ldots, i \ldots\}$. Following standard notation, $\sigma(i)$ denotes the rank of item i, $\sigma^{-1}(j)$ the item at rank j in σ. The permutation matrix Σ corresponding to σ has $\Sigma_{ij} = 1$ iff $\sigma(i) = j$. For any two permutations σ, σ' over $\mathbb{N}^*$, the matrix product $\Sigma\Sigma'$ corresponds to the function composition of permutations $\sigma'(\sigma)$. In our case, Σ, Σ' will be infinite matrices with exactly one 1 in every row and column.

A top-t ordering π^{-1} is the prefix $(\pi^{-1}(1) \ldots \pi^{-1}(t))$ of some infinite ordering. The notation π^{-1} indicates that we observe items, not ranks. For the rest of this paper, the term *ordering* will denote the inverse of a permutation, i.e. the list of items associated to ranks

1, 2, ... In general, a Greek letter π or σ denotes a permutation, also called *ranking*, while the symbols π^{-1}, σ^{-1} denote the corresponding orderings. Further, our notation distinguishes when possible between observed orderings, denoted by π^{-1}, which by virtue of being observed, are always truncated, and the "central permutations", ideal infinite objects denoted by σ. What we try to estimate is a top-t' ordering of σ and this is denoted by σ^{-1}.

The matrix Π of a top-t ordering π has t columns, each with an infinite number of zeros and with a 1 in row $\pi^{-1}(j)$, $j = 1 : t$. Rigourously this matrix should be denoted Π^T since it is the matrix of π^{-1} but we opt for Π to simplify notation. For a permutation σ and a top-t ordering π^{-1}, the matrix $\Pi^T\Sigma$ corresponds to the composition $\sigma(\pi^{-1})$ listing the ranks in σ of the items in π^{-1}. We shall use its transpose, $\Sigma^T\Pi$ which is the $\infty \times t$ matrix formed with rows $\pi^{-1}(1)\dots\pi^{-1}(j)$ of Σ as columns. For any $\Sigma^T\Pi$ matrix, its *code* $(s_j,\, j = 1 : t)$ is defined as follows: $1 + s_j$ is the rank of $\pi^{-1}(j)$ in $\sigma_{|\mathbb{N}^*\setminus\{\pi^{-1}(1),\dots\pi^{-1}(j-1)\}}$. Thus, s_1 is the number of 0's preceding the 1 in the first column of $\Sigma^T\Pi$; after we delete the row containing this 1, s_2 is the number of 0's preceding the 1 in the second column; after deleting the row containing this 1, s_3 is the number of 0's preceding the 1 in the third column, and so on. Hence $s_j \in \{0, 1, 2, \dots\}$,

$$s_j(\Sigma^T\Pi) \;=\; \sigma(\pi^{-1}(j)) - 1 - \sum_{j'<j} 1_{[\sigma(\pi^{-1}(j'))<\sigma(\pi^{-1}(j))]}$$

We now introduce the distance

$$d_\theta(\pi^{-1}, \sigma) \;=\; \sum_{j=1}^{t} \theta_j s_j(\Sigma^T\Pi) \qquad (1)$$

with $\theta = (\theta_{1:t})$ a vector of *strictly positive* parameters; d_θ is the extension to infinite orderings of the d_θ of [6]. In general, this "distance" between a top-t ordering and an infinite ordering is not a metric. When θ_j are all equal $d_\theta(\pi^{-1}, \sigma) = \theta d_K(\pi^{-1}, \sigma)$, with the d_K known as the Kendall distance. d_K counts the number of adjacent transpositions needed to make σ compatible with π^{-1} and is a metric.

3 The Infinite Generalized Mallows model

Now we are ready to introduce the *infinite generalized Mallows (IGM)* model. We start with the observation that as any top-t ordering can be represented uniquely by a sequence of t natural numbers, defining a distribution over the former is equivalent to defining a distribution over the latter, which is a more intuitive task. In keeping with the GM paradigm, we shall choose t-sequences for which each s_j is sampled independently from a discrete exponential with parameter $\theta_j > 0$.

$$P(s_j) \;=\; \frac{1}{\psi(\theta_j)} e^{-\theta_j s_j}, \qquad s_j \;=\; 0,\, 1,\, 2,\, \dots \qquad (2)$$

The normalization constant is $\psi(\theta_j) = \sum_{k=0}^{\infty} e^{-\theta_j k} = \frac{1}{1-e^{-\theta_j}}$. The IGM model can now be defined as

$$P_{\theta,\sigma}(\pi^{-1}) \;=\; e^{-\sum_{j=1}^{t}[\theta_j s_j(\Sigma^T\Pi) + \ln\psi(\theta_j)]} \qquad (3)$$

The distribution $P_{\theta,\sigma}$ has a t-dimensional real parameter θ and an infinite-dimensional discrete parameter σ. Any top-t ordering π^{-1} stands for a set of infinite sequences starting with $s_{1:t}$, $P_{\theta,\sigma}$ can be viewed as the marginal of $s_{1:t}$ in the infinite product space defined by the distribution $P_{\theta,\sigma}(s) \;=\; e^{-\sum_{j=1}^{\infty}[\theta_j s_j + \ln\psi(\theta_j)]}$, $s \in \mathbb{N}\times\mathbb{N}\times\dots$. In contrast with the finite GM, the parameters θ_j must be strictly positive in order for the probability distribution to exist. The most probable π^{-1} for any given t is the top-t ordering which corresponds to $s_1 = \dots = s_t = 0$. This is the top-t prefix of σ^{-1}.

The ordering σ is called the *central permutation* of $P_{\theta,\sigma}$. The parameters θ control the spread around the mode σ. Larger θ_j correspond to more concentrated distributions. These facts are direct extensions of statements about the GM model from [6] and therefore the detailed proofs are omitted.

4 Estimating the model from data

We are given a set of N top-t orderings $\mathcal{D}$. Each $\pi^{-1} \in \mathcal{D}$ can have a different length t; all π^{-1} are sampled independently from a $P_{\theta,\sigma}$ with unknown parameters. We propose to estimate θ, σ from this data in the ML paradigm. We will start by rewriting the log-likelihood of the model, in a way that will uncover a set of sufficient statistics. Then we will show how to estimate the model based on the sufficient statistics.

4.1 Sufficient statistics

The following result lets us understand the structure of the log-likelihood and is thus key to the discrete optimization over σ.

For any square (infinite) matrix $A \in \mathbb{R}^{\mathbb{N}^*\times\mathbb{N}^*}$, denote by $L(A)$ the sum of the elements below the diagonal of A, i.e in the *lower triangle* of A. Let $L_\sigma(A) = L(\Sigma^T A\Sigma)$, and let $\mathbf{1} \in \mathbb{R}^{\mathbb{N}^*}$ be the vector of all 1's. For any π^{-1} let t_π be its length and denote $t_{max} = \max_{\mathcal{D}} t_\pi$, $T = \sum_{\mathcal{D}} t_\pi$.

Theorem 1

$$\ln P_{\theta,\sigma}(\mathcal{D}) = -\sum_{j\geq 1}[\theta_j L_\sigma(R_j) + N_j \ln\psi(\theta_j)] \qquad (4)$$

where $R_j = q_j\mathbf{1}^T - Q_j$*, and* N_j *is the number of* $\pi^{-1} \in \mathcal{D}$ *that have length* $t_\pi \geq j$ *(in other words, that contain rank* j*);* $q_j = [q_{i,j}]_{i\in\mathbb{N}^*}$*, with* $q_{i,j}$ *being the number of times* i *is observed in rank* j *in the data* $\mathcal{D}$*,* $Q_j = [Q_{ii',j}]_{i',i\in\mathbb{N}^*}$ *is a matrix whose element* $Q_{ii',j}$ *counts how many times* $\pi(i) = j$ *and* $\pi(i') < j$*.*

Proof Let Q_0 be the infinite matrix that has 1 above the main diagonal and 0 elsewhere, $(Q_0)_{ij} = 1$ iff $j > i$ and let $\Pi_{:j}$ denote the j-th column of Π. By definition, s_j represents the number of 0's preceding 1 in column j, minus all the 1's in the submatrix $\Sigma^T\Pi_{1:\sigma(\pi^{-1}(j)),1:j}$, i.e $s_j(\Sigma^T\Pi) = \sum_{l\geq 1}(Q_0\Sigma^T\Pi_{:j})_l(\mathbf{1} - \Sigma^T\Pi_{:1} - \Sigma^T\Pi_{:2} - \ldots\Sigma^T\Pi_{:j-1})_l = (\mathbf{1} - \sum_{j'<j}\Sigma^T\Pi_{:j'})^TQ_0\Sigma^T\Pi_{:j}$
$= \mathbf{1}^TQ_0\Sigma^T\Pi_{:j} - \sum_{j'<j}\Pi_{:j'}^T\Sigma Q_0\Sigma^T\Pi_{:j}$
$= \operatorname{trace} Q_0\Sigma^T[\Pi_{:j}\mathbf{1}^T - \sum_{j'<j}\Pi_{:j}\Pi_{:j'}^T\Sigma]$
$= L(\Sigma^T[\Pi_{:j}\mathbf{1}^T - \sum_{j'<j}\Pi_{:j}\Pi_{:j'}^T]\Sigma) = L_\sigma(\Pi_{:j}\mathbf{1}^T - \sum_{j'<j}\Pi_{:j}\Pi_{:j'}^T)$ In the first equality we use the fact that multiplying left by Q_0 counts the zeros preceding 1 in a column, and in the last we use the identity $\mathbf{1}^T\Sigma = \mathbf{1}^T$. Because L_σ is a linear operator, summing over $\pi^{-1} \in \mathcal{D}$ yields $\sum_{\pi^{-1}\in\mathcal{D}} s_j(\Sigma^T\Pi) = L_\sigma(\left[\sum_{\pi^{-1}\in\mathcal{D}}\Pi_{:j}\right]\mathbf{1}^T - \left[\sum_{\pi^{-1}\in\mathcal{D}}\sum_{j'<j}\Pi_{:j}\Pi_{:j'}^T\right])$. It is easy to verify now that the first sum represents q_j and the second one represents Q_j. By setting $R_j(\pi^{-1}) = Q_j(\pi^{-1}) = 0$ for $j > t_{max}$, we can write the log-likelihood as in (4). □

Corollary 2 *If* $\theta_1 = \theta_2 = \ldots = \theta$ *then the log-likelihood of the data* $\mathcal{D}$ *can be written as*

$$\ln P_{\theta,\sigma}(\mathcal{D}) = -\theta L_\sigma(R) - T\ln\psi(\theta) \qquad (5)$$

with $R = q\mathbf{1}^T - Q$ *and* $q = \sum_j q_j$*,* $Q = \sum_j Q_j$*.*

Note that q_i, $Q_{ii'}$ represent respectively the number of times item i is observed in the data and the number of times item i' precedes i in the data.

Theorem 1 and its corollary 2 show that the infinite model $P_{\theta,\sigma}$ has *sufficient statistics*. As θ, σ of the model are infinite, the sufficient statistics R_j, N_j (or R) are infinite too. However, for any finite data set, these matrices and vector will contain only a finite number of non-zero entries. Another consequence is that the data will only constrain a finite number of parameters of the model. The log-likelihood (4) depends only on the parameters $\theta_{1:t_{max}}$. Maximizing likelihood will determine $\theta_{1:t_{max}}$ leaving the other θ_j parameters undetermined.

Let n be the number of distinct items observed in the data. From σ, we can estimate at most its restriction to the items observed, i.e. the restriction of σ to the set $\bigcup_{\mathcal{D}}\pi^{-1}$. In other words the data determine a top-n (partial) ordering corresponding to σ^{ML}, leaving the rest undetermined[1].

Note also that, unlike in the finite case, the sufficient statistics do not necessarily compress the data. They can take up more space than storing the permutations, and their size grows with N.

4.2 ML estimation: the case of a single θ

We now go on to the practical estimation of θ and σ starting with the case of equal θ_j, i.e $\theta_1 = \theta_2 = \ldots = \theta$. In this case, equation (5) shows that the estimation of θ and σ decouple. For any fixed σ, equation (5) attains its minimum at

$$\theta = \ln(1 + T/L_\sigma(R)). \qquad (6)$$

In contrast to the above simple formula, for the finite GM, the likelihood has no analytic solution for θ [6]. The estimated value of θ increases when $L_\sigma(R)$ decreases. In other words, if the lower triangle of $\Sigma^TR\Sigma$, counting the "out of order" pairs, has very low counts, we conclude that the distribution is very concentrated, hence has a high θ.

Estimating σ^{ML} amounts to minimizing $L_\sigma(R)$ w.r.t σ, independently of the value of θ. The optimal σ according to Corollary 2 is the (partial) permutation that minimizes the lower triangular part of $\Sigma^TR\Sigma$[2]. To find it we exploit an idea first introduced in [11]. This idea is to search for $\sigma^{-1} = (i_1, i_2, i_3, \ldots)$ in a stepwise fashion, starting from the top item i_1 and continuing down.

Assuming for a moment that $\sigma^{-1} = (i_1, i_2, i_3, \ldots)$ is known, the cost to be minimized $L_\sigma(R)$ can be decomposed columnwise as $L_\sigma(R) = \sum_{l\neq i_1}R_{li_1} + \sum_{l\neq i_1,i_2}R_{li_2} + \sum_{l\neq i_1,i_2,i_3}R_{li_3} + \ldots$ where the number of non-trivial terms is one less than the dimension of R. It is on this decomposition that the search algorithm is based. If σ^{-1} is not known, a search algorithm could try every i_1 in turn, saving the partial sums, then for a chosen i_1 value could try all i_2's that could follow it, etc. This type of search is represented by a *search tree*, whose nodes are candidate prefixes for σ^{-1}.

[1] That not even the restriction to the observed items is always completely determined can be seen by the following example. Assume the data consists of the the two top-t orderings (a, b, c), (a, b, d). Then (a, b, c, d) and (a, b, d, c) are both ML estimates for σ^{-1}; hence, it would be more accurate to say that the ML estimate of σ^{-1} is the *partial ordering* $(a, b, \{c, d\})$.

[2] If the optimum is a partial permutation $\hat{\sigma^{-1}}$ then any permutation compatible with $\hat{\sigma^{-1}}$ will be a minimizer of $L_\sigma(R)$.

For our problem, the search tree has $n!$ terminal nodes, one for each possible ordering of the observed items. Finding the lowest cost path through the tree is equivalent to minimizing $L_\sigma(R)$. *Branch-and-bound (BB)* [12] algorithms are methods to explore the tree nodes in a way that guarantees that the optimum is found, even though the algorithm may not visit all the nodes in the tree. The number of nodes explored in the search for σ^{-1} depends on the sufficient statistics matrix R. In the worst case, the number of nodes searched can be a significant fraction of $n!$ and as such intractable for all but small n. However, if the data distribution is concentrated around a mode, then the search becomes tractable.

We call the BB algorithm for estimating σ the BBoundR. The algorithm's main steps, characteristics of a BB algorithm, are given in figure 1. In addition to the exact BBoundR algorithm, various heuristic search techniques can be used. Two of them which showed good performance for the standard GM model are the greedy (depth-first) search and the heuristic of [7]. In the latter, one obtains σ by sorting $r_l = \sum_k R_{kl}$ in increasing order[3].

In conclusion, to estimate the parameters from data in a single parameter case, one first computes the sufficient statistics, then a prefix of σ^{-1} is estimated by BBoundR or heuristic methods, and finally, with the obtained ordering of the observed items, one can compute the estimate of θ.

4.3 ML estimation: the case of general θ

Maximizing the likelihood of the data $\mathcal{D}$ is equivalent, by Theorem 1, with minimizing

$$J(\theta, \sigma) \;=\; \sum_j [\theta_j L_\sigma(R_j) + N_j \ln \psi(\theta_j)] \qquad (7)$$

This estimation equation does not decouple w.r.t θ and σ. Minimization is however possible, due to the following two observations. First, for any fixed set of θ_j values, minimization w.r.t σ is possible by the algorithms described in the previous section. Second, for fixed σ, the optimal θ_j parameters can be found analytically by $\theta_j = \ln(1 + N_j/L_\sigma(R_j))$.

The two observations immediately suggest an alternate minimization approach to minimizing J. The algorithm is given in Figure 2. As both steps increase the likelihood, the algorithm will stop in a finite number of steps at a local optimum

[3]Described here is a simplified version of the SortRowsR heuristic. The algorithm as proposed by [7] also performs limited search around the obtained permutation.

Algorithm BBoundR

Input Matrix R

The algorithm maintains a priority queue Q storing search tree nodes. For each node ρ, we store: the path to the node ($r_{1:j}$), the cost C of this path, a lower bound A on the cost-to-go, and the sum $T = C + A$. Nodes are prioritized by T.

While Q is not empty:

1. $\rho \leftarrow$ Extract $\min(Q)$
2. if $j(\rho) = n - 1$ **Output** ρ. **Stop**
3. else, for $k = 1 : n - j$
   (a) Create child ρ' of ρ
   (b) Evaluate $C(\rho'), A(\rho'), T(\rho')$
   (c) enqueue ρ' in Q

Figure 1: Algorithm BBoundR.

Algorithm EstimateSigmaTheta

Input Sufficient statistics R_j, N_j, $j = 1 : t_{max}$
Initial parameter values $\theta_{1:t_{max}} > 0$

1. Iterate until convergence:
   (a) Calculate $R_\theta = \sum_j \theta_j R_j$
   (b) Find partial ordering $\sigma^{-1} = \text{argmin}_\sigma L_\sigma(R_\theta)$ by BBoundR
   (c) Estimate $\theta_j = \ln[1 + N_j/L_\sigma(R_j)]$ for $j = 1 : t_{max}$

Output σ^{-1}, $\theta_{1:t_{max}}$

Figure 2: Algorithm EstimateSigmaTheta.

5 Clustering

Having defined a distance and a method for estimating ML parameters gives one access to a large number of the existing clustering paradigms originally defined for Euclidean data. For instance, the extensions of the K-means and EM algorithms to infinite orderings is immediate, and so are extensions to other distance-based clustering methods. In addition, we introduce a nonparametric clustering method, the *Exponential Blurring Mean-Shift (EBMS)*. Nonparametric clustering is motivated by the fact that in many real applications the number of clusters is unknown and outliers exist. We adapt for ranked data the well known blurring mean-shift algorithm [2]. We choose the exponential kernel $K_\theta(\pi, \sigma) = e^{-\theta d_K(\pi,\sigma)}$. For a data set of top-$t$ rankings, the Nadaraya-Watson kernel estimator

Table 1: Results of estimation experiments. Top: mean and standard deviation of θ^{ML} for two values of the true θ and for different t values and sample sizes n. Bottom: the proportion of cases when the ordering error, i.e the number inversions w.r.t the true σ^{-1} was 0, respectively 1. Each estimation was replicated 25 or more times.

Estimates of θ (mean stdev)							
θ	**N**	200		500		2000	
		mean	std	mean	std	mean	std
0.69	$t=2$	0.68	0.04	0.68	0.03	0.68	0.024
	$t=4$	0.67	0.03	0.69	0.02	0.69	0.01
	$t=8$	0.68	0.02	0.69	0.01	0.69	0.007
1.38	$t=2$	1.34	0.13	1.37	0.09	1.37	0.04
	$t=4$	1.40	0.06	1.38	0.05	1.38	0.03
	$t=8$	1.37	0.03	1.38	0.03	1.38	0.01

Ordering error							
θ	**N**	200		500		2000	
		$d_K = 0$	$d_K = 1$	$d_K = 0$	$d_K = 1$	$d_K = 0$	$d_K = 1$
0.69	$t=2$	0.36	0.38	0.32	0.38	0.24	0.36
	$t=4$	0.28	0.46	0.36	0.28	0.32	0.42
	$t=8$	0.40	0.38	0.36	0.34	0.40	0.30
1.38	$t=2$	0.78	0.18	0.76	0.18	0.78	0.18
	$t=4$	0.80	0.20	0.82	0.14	0.80	0.16
	$t=8$	0.84	0.14	0.72	0.22	0.92	0.08

is $\widehat{r}(\pi_i) = \sum_{j=1}^{n} \frac{\exp^{-\theta d(\pi_i,\pi_j)}}{\sum_{i=1}^{n} \exp^{-\theta d(\pi_i,\pi_j)}} \pi_j$.

The EBMS algorithm is summarized in Figure 3. It shift the "points" (i.e top-t orderings) to new locations obtained by a locally weighted combination of all the data. Thus, every π^{-1} is "attracted" towards its closest neighbors; as the shifting is iterated the data collapse into one or more clusters. The *scale parameter* θ influences the size of the local neighborhood of a top-t ordering, and thereby controls the granularity of the final clustering: for small θ values (large neighborhoods), points will coalesce more and few large clusters will form; for large θ's the orderings will cluster into small clusters and singletons. In the EBMS algorithm, we estimate the scale parameter θ at each iteration by solving the equation in step (d). The l.h.s of this equation represents the average distance in the data set, while the r.h.s is the expected distance to the centroid under the Infinite GM model for permutations of length t_π.

In step 5c of the algorithm, the new ranking can be much longer than the original partial ranking. As will be shown in section 7, the last ranks are subject to noise and overfitting. Therefore we truncate the new ranking back to t.

Since at each step we round $\widehat{r}(\pi_i)$ to the closest permutation, the algorithm will stop in a finite number of steps, when no ordering moves from its current position.

In the algorithm one evaluates distances between top-

Algorithm EBMS

Input Top-t orderings $\mathcal{D} = \{\pi_i^{-1}\}_{i=1:N}$ with same length t_π.

1. Count the distinct permutations to obtain reduced $\tilde{\mathcal{D}}$ and counts $n_i \geq 1$ for each ordering $\pi_i^{-1} \in \tilde{\mathcal{D}}$.
2. For $\pi_i^{-1} \in \tilde{\mathcal{D}}$ compute q_i, Q_i, R_i the sufficient statistics of a single data point.
3. For $\pi_i^{-1}, \pi_j^{-1} \in \tilde{\mathcal{D}}$ calculate Kendall distance $d_{ij} = d_K(\pi_i^{-1}, \pi_j^{-1})$
4. Set the scale θ by solving the equation

$$\frac{2}{\tilde{N}(\tilde{N}-1)} \sum_{i<j} d_{ij} = \frac{t_\pi \times e^{-\theta}}{1-e^{-\theta}} - \sum_{j=1}^{t_\pi} \frac{j \times e^{-\theta}}{1-e^{-j\theta}}$$

5. For $\pi_i \in \tilde{\mathcal{D}}$ *(Compute weights and shift)*
   (a) For $\pi_j \in \tilde{\mathcal{D}}$: set $\alpha_{ij} = \frac{\exp(-\theta d_{ij})}{\sum_{j'=1}^{n} \exp(-\theta d_{ij'})}$
   (b) Calculate $\bar{R}_i = \sum_{\pi_j \in \tilde{\mathcal{D}}} n_j \alpha_{ij} R_j$
   (c) Estimate σ_i^{-1} the "central" permutation that optimizes $\bar{R}_i$ by BBOUNDR or by heuristics
   (d) Set $\pi_i^{-1} \leftarrow \sigma_i^{-1}(1:t_\pi)$
6. Repeat from step 1 until no π_i^{-1} changes.

Output $\tilde{\mathcal{D}}$

Figure 3: The EBMS algorithm.

t orderings. We define the distance $d(\pi, \pi')$ to be the distance between the sets of infinite orderings compatible with π^{-1} respectively $(\pi')^{-1}$. We need to extend the Kendall distance to cover this case and for this we adopt an idea from [4] which lets us express and evaluate $d(\pi, \pi')$ efficiently. The details were omitted for lack of space but they are available in the full paper [10].

6 The conjugate prior

The existence of sufficient statistics implies the existence of a conjugate prior for the parameters of model (3) [5]. Here we introduce the general form of this prior and show that computing with the conjugate prior (or posterior), is significantly harder than computing with the likelihood (3).

We shall assume for simplicity that all top-t rankings have the same t. Consequently, our parameter space consists of the real positive vector $\theta_{1:t}$ and the discrete infinite parameter Σ. Let ν denote the *prior strength*, representing the equivalent sample size, and

λ_1, Λ_j, $j = 2:t$ be the prior parameters corresponding to the sufficient statistics q_1, $Q_{2:t}$, normalized.

Proposition 3 *Let $\nu > 0$, λ_1 be a vector and Λ_j, $j = 2:t$ denote a set of possibly infinite matrices satisfying $\lambda_1 \geq 0$; $\Lambda_{ii',j} \geq 0$, for all i, i', j; $\mathbf{1}^T \Lambda_j \mathbf{1} = (j-1)$ for all $j > 1$. Denote $\mathbf{\Lambda} = \{\nu, \lambda_1, \Lambda_{2:t}\}$ and $R_j^0 = \Lambda_j \left(\frac{1}{j-1}\mathbf{1}\mathbf{1}^T - I\right)$ for $j > 1$, $R_1^0 = \lambda_1 \mathbf{1}^T$. Then, the distribution*

$$P_{\mathbf{\Lambda}}(\sigma, \theta) \;\propto\; e^{-\nu \sum_{j=1}^{t} [\theta_j L(\Sigma^T R_j^0 \Sigma) + \ln \psi(\theta_j)]} \qquad (8)$$

is a conjugate prior for the model $P_{\theta,\sigma}(\pi^{-1})$ in (3).

Proof Given observed permutations $(\pi^{-1})_{1:N}$, the posterior distribution of (σ, θ) is updated by $P(\theta, \sigma \mid \mathbf{\Lambda}, (\pi^{-1})_{1:N}) \propto \exp\left(-\sum_{j=1}^{t} [(\nu L_\sigma(R_j^0) + L_\sigma(R_j))\theta_j + (N+\nu)\ln\psi(\theta_j)]\right)$ $\exp\left(-(N+\nu)\sum_{j=1}^{t}[\theta_j L_\sigma\left(\frac{\nu R_j^0 + R_j}{N+\nu}\right) + \ln\psi(\theta_j)]\right)$. If the hyperparameters $\nu, \lambda_1, \Lambda_{2:t}$ satisfy the conditions of the proposition, then the new hyperparameters $\mathbf{\Lambda'} = \{\nu + N, (\nu\lambda_1 + q_1)/(\nu + N), (\nu\Lambda_j + R_j)/(\nu + N), j = 2:t\}$ satisfy the same conditions. □.

The conjugate prior is defined in (8) only up to a normalization constant. This normalization constant is not always computable in closed form. Another aspect of conjugacy is that one prefers the conjugate hyperparameters to represent expectations of the sufficient statistics under some $P_{\theta,\sigma}$. The conditions in Proposition 3 are necessary, but not sufficient to ensure this fact.

Proposition 4 *Let $P_{\mathbf{\Lambda}}(\sigma, \theta)$ be defined as in (8) and $S_j^* = L_\sigma(\nu R_j^0 + R_j)$. Then,*

$$P_{\mathbf{\Lambda}}(\theta_j \mid S_j^*) \;=\; Beta_{S_j^*, \nu+1}(e^{-\theta_j}) \qquad (9)$$

$$P_{\mathbf{\Lambda}}(\sigma) \;\propto\; \prod_{j=1}^{t} Beta(S_j^*(\sigma), 1+\nu) \qquad (10)$$

where $Beta_{\alpha,\beta}$ denotes the Beta distribution and $Beta(x, y)$ denotes Euler's Beta function.

Proof sketch Replacing $\psi(\theta_j)$ with its value yields

$$P_{\mathbf{\Lambda}}(\theta_j \mid \sigma) \;\propto\; e^{-S_j^* \theta_j}(1 - e^{-\theta_j})^{N+\nu} \qquad (11)$$

From which the desired results follow. □

We have shown thus that closed form integration over the continuous parameters θ_j is possible. This result is entirely new, as no analog result, and no closed form integration is possible for the GM model with n finite.

The exact summation over the discrete parameters poses much harder challenges, described in the full paper [10], and is still an open problem.

7 Experiments

7.1 Estimation experiments, single θ

In these experiments we generated data from an infinite GM model with constant $\theta_j = \ln 2, \ln 4$ and estimated the central permutation and the parameter θ. To illustrate the influence of t, t_π was constant over each data set. The results are summarized in Table 1.

Note the apparent lack of convergence of the σ^{ML} estimates. This is due to the fact that, as either N or t increase, n_{items} the number of items ranked increases. The least frequent items will have very little support from the data and will be those misranked. We have confirmed this by computing the distance between the true σ^{-1} and our estimate, restricted to the first t ranks. This was always 0, with the exception of $\overline{\overline{n}} = 200, \theta = 0.69, t = 2$ when it averaged 0.04 (2 cases in 50 runs).

Even so the table shows that most ordering errors are no larger than 1. We also note that the sufficient statistic R is an unbiased estimate of the expected R. Hence, for any fixed length $\tilde{t}$ of σ^{ML}, the σ^{-1} estimated from R should converge to the true σ^{-1} (see also [7]). The θ^{ML} based on the true σ^{-1} is also unbiased and asymptotically normal. Another peculiarity is the "asymmetry" of the error in θ^{ML}. Recall that by equation (6) θ is a decreasing function of $L_\sigma(R)$. If the true σ^{-1} is not optimal for the given R, due to sample variance, then θ^{ML} will tend to overestimate θ. Hence θ^{ML} is a *biased* estimate of θ. If however, due to imperfect optimization, the estimated $(\sigma^{-1})^{ML}$ is not optimal and has higher cost than σ^{-1}, then θ^{ML} will err towards underestimation.

7.2 Estimation experiments, general θ

We now generated data from an Infinite GM model with $\theta_1 = \ln 2$ or $\ln 4$ and $\theta_j = 2^{-(j-1)/2}\theta_1$ for $j > 1$. As before, t_π was fixed in each experiment at the values 2, 4, 8. As the estimation algorithm has local optima, we initialized the θ parameters multiple times. However, we observed that in all the experiments on artificial data, the iterations converged to the same parameter values for all initializations.

Figure 4 shows the estimated values of θ_j for sample sizes $N = 200$ and 2000 and for the case $\theta_1 = 0.69$ (the more dispersed distribution) and $t = 8$. The results of the other experiments are similar, and they are presented in the full paper.

Qualitatively, the results are similar to those for single θ, with the main difference stemming from the fact that, with decreasing θ_j values, the sampling distribution of the data is much more spread, especially w.r.t

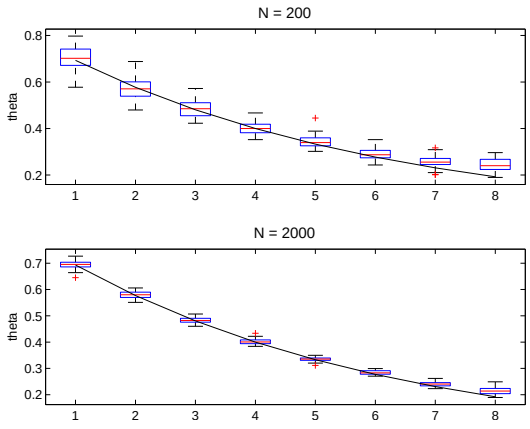


Figure 4: Example of $\theta_{1:t}$ parameters estimation; $t = 8$, sample sizes $N = 200, 2000$. Boxplots over 50 random samples, with j on the horizontal axis. The continuous line crossing the box plots marks the true values of the parameters $\theta_{1:8}$ (exponential decay starting from $\theta_1 = 0.69$).

the lower ranks. Therefore, the bias in θ_j is more pronounced for larger j (and for smaller N and smaller t). For the same reason, the number of observed items n is much larger than before (hundreds vs. less than 20) and consequently the ordering errors w.r.t all the observed items are also much larger[4]. But, if one considers only the top-t part of $(\sigma^{-1})^{ML}$ then the results are as good as those for a single θ, i.e 2 errors in 50 runs for $t = 2, \theta = 0.69, N = 200$ and *zero errors in all other trials*. We have experimented with t up to 22, estimating 22 θ_j parameters, with similar results.

The next experiment was conducted with the data collected by Cohen, Schapire and Singer for their [3] paper. The data consists of a list of 157 universities, the queries, and a set of 21 search engines, the "experts". Each search engine outputs a list of up to $t_{max} = 30$ URL's when queried with the name of the university. The data set provides also a "target" output for each query, which is the university's home page.

Hence, we have 147 estimation problems (10 universities with no data), with sample size $N \leq 21$ (as some experts return empty lists) and with variable length data ranging from $t = 1$ to $t = 30$. Figure 5, a and b give a summary view of number of samples for each rank N_j, $j = 1 : t$, and respectively the total number of items n per query. These values suggest that estimating a fully parameterized model with distinct $\theta_{1:30}$ may lead to overfitting and therefore we estimate several parameterizations, all having the form $\Theta_r = (\theta_1, \theta_2, \ldots \theta_{r-1}, \theta_r, \theta_r, \ldots \theta_r)$. In other words, ranks $1 : r-1$ have distinct parameters, while the remaining ranks share 1 parameter θ_r. We call $\theta_{1:r-1}$ the *free* parameters and θ_r the *tied* parameter. For $r = 1$ we have the single parameter model, and for $r = t_{max} = 30$ we have the fully parameterized model.

Estimating a model with r parameters is done by a simple modification of the EstimateSigmaTheta algorithm which is left to the reader. The estimation algorithm was started from the fixed value $\theta_j = 0.1$ for all runs. The number of iterations to convergence was typically in the range 10–30.

In figure 5 we give a synopsis of the values of the θ parameters under different models. The single parameter models yields θ values in the range [0.007, 0.104] with the 10%, 50% and 90% quantiles being respectively 0.009, 0.018,and 0.032. The parameters θ are on average decreasing in all models, with the free parameters higher than the tied parameters for the remaining ranks. Notice also that for the models with fewer parameters the values of the free parameters tend to be higher than the corresponding values in models with more parameters. Compare for instance the values of θ_1 in the two-parameter model with θ_1 in the 30 parameter model.

For each query and each model size, we computed the rank of the true university home page, i.e the *target*, under the estimated central permutation σ^{ML}. Assuming the search engines are reasonably good, this rank is an indirect indicator of the goodness of a model. In addition, for each query, we selected one model by BIC and calculated the target ranks for these models. The BIC selects predominantly the single parameter model (124 out of 147 cases). Table 2 gives the results.

7.3 Clustering experiments

We sampled orderings from 3 clusters, each an Infinite GM model with a single spread parameters θ, equal to 1.5, 1.0, 0.7 respectively. The cluster centers are random permutations of infinite many objects. A data set contains 150 samples from each cluster, plus 50 outliers. All top-t ordering had the same length t. We experimented with $t_\pi = 4, 6, 8$.

We ran the Exponential Blurring Mean-Shift, K-means, and EM Model-based clustering algorithms 10 times on samples from this distribution. For EBMS, the scale parameter was estimated based on the average of pairwise distances.

For the K-means and model-based algorithms, we experiment with different numbers of clusters, and report the best classification error with respect to the true clustering. This puts these two algorithms at an advantage w.r.t EBMS, but as the results table 3 shows, even so the nonparametric algorithm achieves the best performance.

[4]Complete results are in the full paper [10].

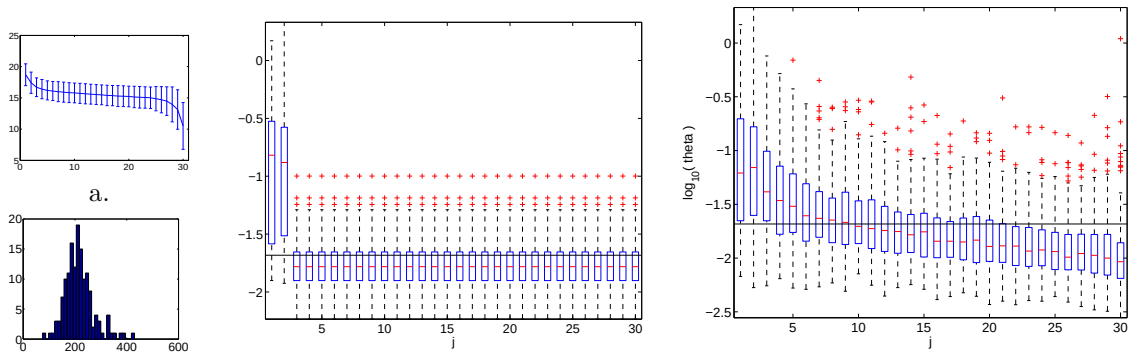


Figure 5: Universities data: mean and standard deviation of the number of samples per rank, over all queries (a); histogram of the number of distinct items observed (b); boxplots of the Θ estimates over all queries for models with 3 and 30 parameters (c,d). The vertical axis of the scale is *logarithmic, base 10*, i.e 0 corresponds to $\theta_j = 1$ and -2 to $\theta_j = 0.01$. For clarity, the distribution of the tied parameter is replicated for $j = r : t_{max}$. The black horizontal line marks the mean value of θ in the single parameter model.

Table 2: Mean and median of the rank of the target web page under each model, and under the BIC selected model. The rank is $t_{max} + 1 = 31$ if the target is not in the search engine's top-t ranking. The statitics are computed once over all 147 universities and once over a subset of 74 universities where the target is always ranked in the first 30; the subset is labeled as "good".

Model size	1	2	3	4	5	6	7	8	9	10	30	BIC
Mean rank (good)	5.3	5.7	4.2	4.2	**4.1**	**4.1**	4.4	4.5	5.0	5.2	5.1	5.37
Median rank (good)	3	3	**1.5**	2	2	2	2	2	2	3	3	2.5
Mean rank (all)	16.5	16.1	**15.4**	15.5	15.5	15.6	15.8	15.7	15.9	16.0	16.0	16.2
Median rank (all)	13	15	11	11	12	**9**	10	10	11	11	11	12

Table 3: Classification Errors: mean and standard deviation of 10 random samples

t	EBMS	K-means	EM
4	**0.0030** (0.0001)	0.1014 (0.0038)	0.1008 (0.0025)
6	**0.0014** (0.0001)	0.0986 (0.0010)	0.1000 (0.0000)
8	**0.0002** (0.0001)	0.0972 (0.0010)	0.1000 (0.0000)

Note that the error rate in table 3 is computed including the outliers, i.e we compared a true clustering with 53 clusters (3 clusters and 50 singletons) to the clustering obtained when the algorithm converged.

For EM and K-means the number of clusters associated with the lowest classification errors was between 3 and 5. From the table we see that the K-means and model based approach identified three primary clusters correctly. K-means did not have the ability of identifying the outliers, so it just assigned each outlier into one of those primary clusters. The model-based approach assigned outliers into primary clusters too, but it also gave more uncertainty on the outliers (the probabilities of outliers belonging to their assigned cluster were relatively smaller than data from primary clusters).

The running time per data set of EBMS was under a minute, and the number of iterations to convergence followed the pattern typical of mean-shift algorithms and was never larger than 10.

8 Related work and discussion

This work acknowledges its roots in the work of Fligner and Verducci on stagewise ordering models [6] and in the recent paper [11]. The latter shows for the first time that GM models have sufficient statistics, and describes an exact but non-polynomial algorithm to find the central permutation. While similarities exist between the algorithm of [11] and the BBoundR algorithm presented here, we stress that our representation (based on the codes s_j) is *different* from the representation (denoted V_j) in [11].

For any given permutation $s_j(\pi) = V_j(\pi^{-1})$. While this difference is trivial for complete permutations, it is not so in the case of missing data. In particular, the distribution of V_j for top-t orderings does not seem to have sufficient statistics for $j > 2$ even in the case of finite permutations. The s_j representation has another advantage that V_j has not: for any finite data set, a parameter s_j is either completely determined or completely undetermined the data, whereas in the reciprocal V_j representation *all* V_j are weakly constrained by data.

While both our BBoundR and the algorithm of [11] perform branch-and-bound search on a matrix of sufficient statistics, the sufficient statistics in the infinite case are derived by an entirely different method, and cannot be obtained by naively replacing the sufficient statistics of the finite case.

The paper [1] uses the s_j representation and outlines its advantages. It is also the first paper to do (EM) clustering of partial orderings, without however recognizing the existence of sufficient statistics. Another interesting application of the GM model to multimodal data is [9] (there the σ's play the role of the data), while a greedy algorithm for consensus ordering with partially observed data is introduced in [3]; [4] is an early work on (Haussdorf) distances for partial orderings. In [8] the authors also introduce an EM algorithm for clustering ranking data for the purpose of analyzing Irish voting patterns. However, the base model used by [8] is not the Mallows model but a model known as Plackett-Luce. The estimation of this from data is much more difficult and, as [8] show, can be only done approximately.

All the above works deal with permutations on finite sets. In fairness to [3] we remark that this work, although it only considers heuristics methods for optimization and introduces a cost function which only later, by [11] is shown to be closely related to the log-likelihood, is motivated by the same problem as ours, i.e dealing with a very large set of items, of which only some are ranked by the "voters".

The paper of [13] studies the space of infinite permutations which differ from the identity in a finite number of positions. In the vocabulary of the present paper, these would be the infinite permutations at finite distance d_K from σ. In a single parameter infinite GM, these infinite permutations are the only ones which have non-zero probability. While from a probabilistic perspective the two views are equivalent, from a practical perspective they are not. We prefer to consider in our sample space all possibile orderings, including those with vanishing probability. It is the latter who are more representative of real experiments. For instance, in the university web sites ranking experiment, our model assumed that there is a "true" central permutation from which the observations were generated as random perturbations. This is already an idealization. But we also have the liberty to assume that the observations are very long orderings which are close to the central permutation only in their highest ranks, and which can diverge arbitrarily far from it in the latter ranks. We consider this a more faithful scenario than assuming in addition that the observations must be identical to the central permutation (and hence to each other!) on all but a finite number of ranks.

We have introduced the first –to our knowledge– stagewise ranking model for infinitely many items. The new probabilistic model has several attractive properties: it handles naturally truncated top-t orderings, it has sufficient statistics, and more importantly we showed that it also has an exact estimation algorithm (albeit intractable in the worst case). As it is known from the study of stochastic models of permutations over finite domains, exact estimation and interpretable parameters are very rare qualities in this field.

Sampling, distance computations, clustering can be performed in this class of models in a natural way and are all tractable. We have paid particular attention to non-parametric clustering by mean-shift blurring, showing by experiments that the algorithm is practical and effective.

An issue not solved for GM models, finite or infinite, is sampling a θ, σ from the conjugate distribution. If this is feasible, one can perform clustering by the DP mixture model, a model-based clustering paradigm widely recognized for its advantages. It is our intention to work in this direction.

Acknowledgments

Thanks to Jon Wellner for bringing up the question of infinite permutations.

References

[1] L. M. Busse, P. Orbanz, and J. Bühmann. Cluster analysis of heterogeneous rank data. In *Proceedings of the International Conference on Machine Learning ICML*.

[2] M. A. Carreira-Perpiñan. Gaussian mean shift is an EM algorithm. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(5):767–776, 2007.

[3] W. C. Cohen, R. S. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[4] D. E. Critchlow. *Metric methods for analyzing partially ranked data*. Number 34 in Lecture notes in statistics. Springer-Verlag, Berlin Heidelberg New York Tokyo, 1985.

[5] M. H. DeGroot. *Probability and Statistics*. Addison–Wesley Pub. Co., Reading, MA, 1975.

[6] M. A. Fligner and J. S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society B*, 48:359–369, 1986.

[7] M. A. Fligner and J. S. Verducci. Multistage ranking models. *Journal of the American Statistical Association*, 88, 1988.

[8] I. Gormley and T. Murphy. Exploring heterogeneity in irish voting data: A mixture modelling approach. *Technical Report*, Department of Statistics, Trinity College Dublin(05/09), 2005.

[9] G. Lebanon and J. Lafferty. Conditional models on the ranking poset. In *Advances in Neural Information Processing Systems*, number 15, Cambridge, MA, 2003. MIT Press.

[10] M. Meila and L. Bao. Estimation and clustering with infinite rankings. Technical Report 529, UW Statistics, 2008.

[11] M. Meilă, K. Phadnis, A. Patterson, and J. Bilmes. Consensus ranking under the exponential model. In R. Parr and L. Van den Gaag, editors, *Proceedings of the 23rd Conference on Uncertainty in AI*, volume 23, page (to appear), 2007.

[12] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley, 1999.

[13] E. Thoma. *Mathematische Zeitschrift*, 1964.

The Phylogenetic Indian Buffet Process: A Non-Exchangeable Nonparametric Prior for Latent Features

Kurt T. Miller
EECS
University of California
Berkeley, CA 94720
tadayuki@cs.berkeley.edu

Thomas L. Griffiths
Psychology and Cognitive Science
University of California
Berkeley, CA 94720
tom_griffiths@berkeley.edu

Michael I. Jordan
EECS and Statistics
University of California
Berkeley, CA 94720
jordan@cs.berkeley.edu

Abstract

Nonparametric Bayesian models are often based on the assumption that the objects being modeled are exchangeable. While appropriate in some applications (e.g., bag-of-words models for documents), exchangeability is sometimes assumed simply for computational reasons; non-exchangeable models might be a better choice for applications based on subject matter. Drawing on ideas from graphical models and phylogenetics, we describe a non-exchangeable prior for a class of nonparametric latent feature models that is nearly as efficient computationally as its exchangeable counterpart. Our model is applicable to the general setting in which the dependencies between objects can be expressed using a tree, where edge lengths indicate the strength of relationships. We demonstrate an application to modeling probabilistic choice.

1 INTRODUCTION

Nonparametric Bayesian analysis provides a way to define probabilistic models in which structural aspects of the model, such as the number of classes or features possessed by a set of objects, are treated as unknown, unbounded and random, and thus viewed as part and parcel of the posterior inference problem. This elegant treatment of structural uncertainty has led to increasing interest in nonparametric Bayesian approaches as alternatives to model selection procedures.

Nonparametric Bayesian methods are based on prior distributions that are defined on infinite collections of random variables—i.e., prior distributions expressed as general stochastic processes. This generality provides a rich language in which to express prior knowledge. In practice, however, the need to integrate over these stochastic processes at inference time imposes a strong constraint on the kinds of models that can be considered. In particular, nonparametric Bayesian models are often based on an assumption of *exchangeability*—the joint probability of the set of entities being modeled by the prior is assumed to be invariant to permutation. A particular example of exchangeability is the "bag-of-words" assumption widely used in the modeling of document collections.

In this paper we aim to extend the range of nonparametric Bayesian modeling by presenting a non-exchangeable prior for a class of nonparametric latent feature models. Our point of departure is the *Indian buffet process* (IBP), a generative process that defines a prior on sparse binary matrices (Griffiths and Ghahramani, 2006). This process, which will be described in more depth later, can be understood through a culinary metaphor in which diners sequentially enter a buffet line and select which dishes to try. As each person moves through the buffet line, they try each previously sampled dish with probability proportional to the number of people who have already tried it. This process can be shown to be exchangeable from the fact that it is obtained as a conditionally-independent set of draws from a Bernoulli process with parameters drawn from an underlying stochastic process known as the beta process (Thibaux and Jordan, 2007).

To obtain a useful non-exchangeable, nonparametric model while retaining the computational tractability of the IBP, we consider a model in which relationships among the diners are expressed by a tree. In this stochastic process—the *Phylogenetic Indian Buffet Process* (pIBP)—the probability that a diner chooses a dish in the buffet line depends not only on the number of previous diners who have chosen that dish, but also on how closely related the current diner is to each of the previous diners. We exploit efficient probabilistic calculations on trees (Pearl, 1988) to obtain a tractable algorithm for taking relatedness into

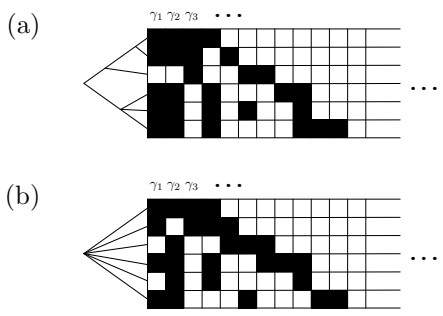


Figure 1: The Phylogenetic Indian Buffet Process. (a) A tree expresses dependencies among featural representations of objects. (b) The Indian Buffet Process is a special case of the pIBP where all branches meet at the root.

account when computing posterior updates under the pIBP prior.

The tree representation is a rich language for expressing non-exchangeability. In particular, factorial and nested models of the kind used in experimental design are readily expressed as trees. Group structure as used in the hierarchical Dirichlet process (Teh et al., 2005) and hierarchical beta process (Thibaux and Jordan, 2007) can also be expressed as trees, as can a variety of other partially exchangeable models (Diaconis, 1988). In biological data analysis we may be able to exploit known phylogenetic or genealogical relationships among species or characters. More generally we may have similarity data available for a set of objects that can be used to build a tree representation. The pIBP uses such representations to induce a prior on featural representations such that objects that are related in the tree will tend to share features (see Figure 1).

It is important to distinguish our approach from previous nonparametric Bayesian work based on random trees (Neal, 2001; Teh et al., 2008). In that work, trees are averaged over and objects remain exchangeable. We are conditioning on a known, fixed tree and our objects are not exchangeable.

While we develop the pIBP in the context of the IBP for concreteness, the idea is actually much broader. The key is that the use of a tree to express relationships among non-exchangeable random variables allows us to exploit the sum-product algorithm in defining the updates for an MCMC sampler. This insight extends the scope of nonparametric Bayesian models without significantly increasing the computational burden associated with inference.

The paper is organized as follows. Section 2 presents a short review of the IBP and then provides a detailed description of the pIBP. Section 3 discusses MCMC inference in models using the pIBP as a prior. Section 4 presents an application of the pIBP to models of human choice, and shows how combining nonparametric methods with a tree-based prior improves performance. Section 5 presents our conclusions.

2 THE PHYLOGENETIC INDIAN BUFFET PROCESS

Griffiths and Ghahramani (2006) proposed the Indian Buffet Process as a prior distribution on sparse binary matrices Z, where the rows of Z correspond to objects and the columns of Z correspond to a set of features or attributes describing these objects. As with other nonparametric Bayesian models, the IBP can be interpreted through a culinary metaphor. In this metaphor, objects correspond to people and features correspond to an infinite array of dishes at an Indian buffet. The first person samples Poisson(α) dishes, where α is a parameter. The i^{th} customer then samples all previously sampled dishes in proportion to the number of people who have already sampled those dishes, and Poisson(α/i) new dishes. This process defines an exchangeable distribution on equivalence classes of Z, the binary matrix with a one at cell (i, k) when the i^{th} customer chooses the k^{th} dish. Figure 1(b) shows a matrix generated from this process.

The IBP can be derived as the infinite limit of a beta-Bernoulli model. Consider a finite model in which there are K features, and let the probability that an object possesses feature k be Bernoulli(π_k). Endowing π_k with a Beta($\alpha/K, 1$) distribution, we obtain the IBP in the limit as $K \to \infty$. The Phylogenetic Indian Buffet Process uses a similar construction. As with the IBP, we will use the term "pIBP" to refer to both the distribution on binary matrices as well as a generative process which induces this distribution. We first describe the distribution on finite matrices at the heart of the pIBP, and then describe the process that results by letting the number of features go to infinity.

2.1 A PRIOR ON FINITE MATRICES

We begin by defining a generative process for Z, an $N \times K$ binary matrix, where N is the number of objects and K is a fixed, finite number of features. We use the following notation. Let z_{ik} denote the (i, k) entry of Z, let z_k be the k^{th} column of Z, let $z_{(-i)k}$ denote the entries of z_k except z_{ik}, and let $Z_{-(ik)}$ denote the entries of the full Z matrix except z_{ik}.

As in the IBP, we associate a parameter π_k to each column, chosen from a Beta($\alpha/K, 1$) prior distribution, where α is a hyperparameter. Given π_k, the marginal

probability that any particular entry in column k is one is equal to π_k. Columns are generated independently. We diverge from the IBP, however, in the specification of the joint probability distribution for the column z_k. In the IBP, the entries of z_k are chosen independently given π_k. In the pIBP, the entries of z_k are dependent, with the pattern of dependence captured by a stochastic process on a rooted tree similar to models used in phylogenetics. In this tree, the N objects being modeled are at the leaves, and lengths are assigned to edges in such a way that the total edge length from the root to any leaf is equal to one. To generate the entries of the k^{th} column, we proceed as follows. Assign the value zero to the root node of the tree. Along any path from the root to a leaf, let this value change to a one along any edge of length t with exponential rate $\gamma_k t$ where $\gamma_k = -\log(1-\pi_k)$. That is, along an edge of length t, let the probability of changing from a zero to a one be $1-\exp(-\gamma_k t)$. Once the value has changed to a one along any path from the root, all leaves below that point are assigned the value one.

The parameterization $\gamma_k = -\log(1-\pi_k)$ is convenient because it ensures that π_k remains the marginal probability that any single feature is equal to one. To see this, simply note that since every leaf node is at distance one from the root, for any entry in the matrix,

$$p(z_{ik} = 1|\pi_k) = 1 - \exp(-(-\log(1-\pi_k))) = \pi_k$$

which also guarantees that we recover the beta-Bernoulli model in the special case where all branches join at the root, as in Figure 1 (b). It is a simple corollary that any set of objects characterized by a set of branches that meet at a single point will be exchangeable within that set, meaning that the tree can be used to capture notions of partial exchangeability.

2.2 CONDITIONAL PROBABILITIES

Now that we have defined the generative model on finite matrices, we show how to evaluate conditional probabilities in this model. We treat the tree as a directed graph with variables at each of the interior nodes and z_{ik} at each leaf i. Then, given π_k, or equivalently γ_k, if there is a length t edge from a parent node x to a child node y, we have

$$\begin{aligned}
p(y=0|x=0,\gamma_k) &= \exp(-\gamma_k t) \\
p(y=1|x=0,\gamma_k) &= 1-\exp(-\gamma_k t) \\
p(y=0|x=1,\gamma_k) &= 0 \\
p(y=1|x=1,\gamma_k) &= 1
\end{aligned}$$

as the conditional probabilities that define a tree-structured graphical model.

Expressing this process as a graphical model makes it possible to efficiently compute various conditional probabilities that are relevant for posterior inference. Specifically, we will need to evaluate

$$p(z_{ik}|z_{(-i)k},\pi_k) \tag{1}$$

for $z_{ik} \in \{0,1\}$ and

$$p(z_{(-i)k}|z_{ik},\pi_k), \tag{2}$$

which are trivial in the beta-Bernoulli model due to the conditional independence of z_{ik}, but more challenging in the pIBP where z_{ik} are no longer conditionally independent. To compute (1), we use the sum-product algorithm (Pearl, 1988). In order to calculate (2), we use the chain rule of probability to get a set of terms similar to (1), the difference being that the posterior in each term is conditional only on a subset of the other variables. Each term can be reduced to a simple sum-product calculation by marginalizing over all variables that do not appear in that term, which can be done easily since all variables appear at the leaves of the tree. Both (1) and (2) can be calculated in $O(N)$ time by a dynamic program.

2.3 A GENERATIVE PROCESS

The pIBP can be described as a sequential generative process that arises when we let K go to infinity in the distribution derived in Section 2.1. This process can again be understood in terms of a culinary metaphor, in which each row of Z is viewed as the choices made by a diner in a buffet line, and in which we specify how each diner chooses their dishes based on the dishes chosen by previous diners. We overview this process here, leaving detailed mathematical derivations for later sections.

Consider a large extended family that is about to choose dishes at a buffet. Assume that we are given a tree describing the genealogical relationships of the family members and assume that dining preferences are related to genealogy. In particular, family members who are more closely related have more similar preferences. Therefore, as each diner moves through the buffet line, their choice of dishes will be more dependent on the selections of previous diners who are closely related to them and less dependent on the selections of other diners.

The pIBP generative process is specified as follows. The first diner (arbitrarily chosen) starts at the head of a buffet line that has infinitely many dishes. This person tries Poisson(α) dishes and also adds a brief annotation to each of these dishes, π_k, drawn uniformly from $[0,1]$. This note, through its previously described equivalent representation, $\gamma_k = -\log(1-\pi_k)$, will allow us to efficiently compute the probability that

subsequent diners choose the k^{th} dish using the sum-product algorithm.

Each subsequent diner enters the buffet line and based on the annotations attached to the dishes as well as the identity of previous diners, samples the k^{th} dish according to the probability (1) where $z_{(-i)k}$ indicates which of the previous diners have chosen the k^{th} dish. Through the stochastic process on the tree, if closely related diners have tried a dish, the current diner is more likely to also sample it. The preferences of all diners who have not entered the buffet line are ignored, which can be done by only performing sum-product on the minimal subtree from the root containing the current diner and all previous diners.

Each diner also samples a number of new dishes. If t_i is the length of the branch connecting diner i to the rest of the minimal subtree just described and $\sum t$ is the total length of the rest of this subtree, then diner i tries Poisson$(\alpha(\psi(\sum t + t_i + 1) - \psi(\sum t + 1)))$ new dishes, where $\psi(\cdot)$ is the digamma function. They also add an annotation, π_k, to each of the new dishes that will be used for future inferences, where the density of π_k is proportional to $\left(1 - (1 - \pi_k)^{t_i}\right)(1 - \pi_k)^{\sum t} \pi_k^{-1}$.

This process repeats until all diners have gone through the buffet line, defining a matrix Z as in the IBP. Though this process is not exchangeable, we can let any family member go first and get the same marginal distribution. This means that each row of Z has a Poisson(α) number of non-zero columns, yielding a sparse matrix as in the IBP. The IBP is the special case of the pIBP corresponding to the tree shown in Figure 1 (b); this fact can be proved by using identities of the digamma function on the integers.

3 INFERENCE BY MCMC

We now consider how to perform posterior inference in models using the pIBP as a prior. As with the IBP, exact inference is intractable, but the model is amenable to approximate inference via Markov chain Monte Carlo (MCMC) (Robert and Casella, 2004). An important point is that even though we are dealing with a potentially infinite number of columns in Z, we only need to keep track of the non-zero columns, a property shared by other nonparametric Bayesian models. Henceforth, let Z refer to all the non-zero columns of the matrix. The fact that the number of ones in any given row has a Poisson distribution means that the number of non-zero columns is finite (with probability one) and generally small.

Given the matrix Z, we assume that data X are generated through a likelihood function $p(X|Z)$. The likelihood may introduce additional parameters that must be sampled as part of the overall MCMC procedure; we will not discuss such parameters in our presentation.

Unlike the IBP, where π_k can always be integrated out, inference in the pIBP requires treating π_k as an auxiliary variable, sampling it when needed and integrating it out when possible. By sampling π_k for non-zero columns of Z as opposed to integrating it out, we are able to exploit the sum-product algorithm as described in Section 2.2. Updating π_k and all z_{ik} for each column takes $O(N + mN)$ time where m is the total number of times a z_{ik} in column k changes value. Once the chain has mixed well m is typically small, so time complexity is only slightly worse than that of the IBP, which is $O(N)$.

Given an initial choice of the non-zero columns of Z and the corresponding π_k for each of these columns, we construct a Markov chain where at each step, we only need to sample each variable from its conditional distribution given all others. We now describe how to sample each of the variables, first considering the variables for "old" columns—those with non-zero entries—and then turning to the addition of "new" columns.

3.1 SAMPLING z_{ik} FOR OLD COLUMNS

The probability of each z_{ik} given all other variables is

$$p(z_{ik}|Z_{-(ik)}, \pi_k, X, \alpha) \propto p(X|Z_{-(ik)}, z_{ik}) p(z_{ik}|z_{(-i)k}, \pi_k), \quad (3)$$

where the first term is the probability of X given a full assignment of the parameters and depends on the specific model being used. The term $p(z_{ik}|z_{(-i)k}, \pi_k)$ can be computed efficiently using the sum-product algorithm as described in Section 2.2. By appropriately caching messages from the sum-product algorithm, this evaluation can be reduced to $O(1)$ time. We evaluate (3) for $z_{ik} = 0$ and $z_{ik} = 1$ and sample z_{ik} from the corresponding posterior distribution. If the value of z_{ik} changes, we then update the messages for sum-product in $O(N)$ time. If a column ever becomes entirely zero, we drop it from Z.

3.2 SAMPLING π_k FOR OLD COLUMNS

We only sample π_k for the old columns of Z, a fact that will be useful in subsequent calculations. The posterior distribution of each π_k is independent of all other π_k and depends only on the k^{th} column of Z.

When resampling π_k, let z_{ik} be a non-zero entry in the k^{th} column. Then,

$$p(\pi_k|z_k, \alpha) \propto p(z_{(-i)k}|\pi_k, z_{ik}) p(\pi_k|z_{ik}, \alpha). \quad (4)$$

Section 2.2 describes how to compute $p(z_{(-i)k}|\pi_k, z_{ik})$ efficiently in $O(N)$ time using the sum-product algo-

rithm. In order to obtain $p(\pi_k|z_{ik},\alpha)$, we compute

$$\begin{aligned} p(\pi_k|z_{ik},\alpha) &\propto \underbrace{p(z_{ik}|\pi_k)}_{\sim\text{Bernoulli}(\pi_k)}\underbrace{p(\pi_k|\alpha)}_{\sim\text{Beta}(\frac{\alpha}{K},1)} \\ &\propto \text{Beta}\left(1+\frac{\alpha}{K},1\right) \\ &\overset{K\to\infty}{\to} \text{Beta}(1,1) = \mathbb{1}_{\{\pi_k\in[0,1]\}}. \end{aligned}$$

We see that we can evaluate $p(\pi_k|z_k,\alpha)$ up to a normalizing constant efficiently, so we can sample the new π_k using a Metropolis-Hastings step.

Specifically, given a proposed value of π' for π_k, the ratio of the posterior probabilities of π' and π_k is

$$\begin{aligned} \frac{p(\pi'|z_k,\alpha)}{p(\pi_k|z_k,\alpha)} &= \frac{p(z_{(-i)k}|\pi',z_{ik})p(\pi'|z_{ik},\alpha)}{p(z_{(-i)k}|\pi_k,z_{ik})p(\pi_k|z_{ik},\alpha)} \\ &= \frac{p(z_{(-i)k}|\pi',z_{ik})}{p(z_{(-i)k}|\pi_k,z_{ik})}\mathbb{1}_{\{\pi'\in[0,1]\}}, \end{aligned}$$

so if we use $q(\pi'|\pi_k)$ as the proposal distribution, then the Metropolis-Hastings acceptance ratio is

$$\min\left[1,\frac{q(\pi_k|\pi')}{q(\pi'|\pi_k)}\frac{p(z_{(-i)k}|\pi',z_{ik})}{p(z_{(-i)k}|\pi_k,z_{ik})}\mathbb{1}_{\{\pi'\in[0,1]\}}\right]. \quad (5)$$

There are many options for q. In our experiments, we used $q(\pi'|\pi_k)\sim\mathcal{N}(\pi_k,\sigma_k^2)$ where $\sigma_k^2 = c\cdot\pi_k(1-\pi_k)+\delta$ with $c=0.06$ and $\delta=0.08$.

3.3 SAMPLING THE NEW COLUMNS

In addition to sampling the values of z_{ik} in old columns, we need to consider the possibility that one of the infinitely many all-zero columns has a single entry that becomes a one. As mentioned in Section 3.2, we only sample values of π for non-zero columns, so when sampling new columns, we must integrate out π.

For finite K, we can directly compute the probability $p(z_{ik}=1|z_{(-i)k}=0)$ for each all-zero column. If t_i is the length of the edge that ends at the i^{th} object and $\sum t$ is the total length of all other edges in the tree, then we get

$$\begin{aligned} &p(z_{ik}=1|z_{(-i)k}=0) \\ &\quad\propto \int_0^1 p(z_{ik}=1|z_{(-i)k}=0,\pi)p(z_{(-i)k}=0|\pi)p(\pi)d\pi \\ &\quad\propto \int_0^1 (1-(1-\pi)^{t_i})(1-\pi)^{\sum t}\pi^{\alpha/K-1}d\pi \\ &\quad= \frac{\Gamma(\alpha/K)\,\Gamma(\sum t+1)}{\Gamma(\sum t+\alpha/K+1)} - \frac{\Gamma(\alpha/K)\,\Gamma(\sum t+t_i+1)}{\Gamma(\sum t+t_i+\alpha/K+1)} \end{aligned}$$

where $\Gamma(\cdot)$ is the gamma function and similarly

$$p(z_{ik}=0|z_{(-i)k}=0)\propto\frac{\Gamma(\alpha/K)\,\Gamma(\sum t+t_i+1)}{\Gamma(\sum t+t_i+\alpha/K+1)}$$

which gives us

$$\begin{aligned} &p(z_{ik}=1|z_{(-i)k}=0) \\ &\quad= 1-\frac{\Gamma(\sum t+t_i+1)}{\Gamma(\sum t+1)}\frac{\Gamma(\sum t+\alpha/K+1)}{\Gamma(\sum t+t_i+\alpha/K+1)}. \end{aligned}$$

Therefore, the event that we sample $z_{ik}=1$ in any particular all-zero column is Bernoulli with the above probability. Treating the value α/K as a variable in the equation for $p(z_{ik}=1|z_{(-i)k}=0)$ and doing a first-order Taylor expansion about zero, we get

$$\begin{aligned} &p(z_{ik}=1|z_{(-i)k}=0) \\ &\quad= \frac{\alpha}{K}\left(\psi\left(\sum t+t_i+1\right)-\psi\left(\sum t+1\right)\right)+o\left(\frac{1}{K}\right), \end{aligned}$$

where $\psi(\cdot)$ is the digamma function.

As K grows, the probability that any particular all-zero column becomes non-zero goes to zero. On the other hand, we have a growing number of these Bernoulli variables. Using the fact that $\text{Binomial}\left(K,\frac{\lambda}{K}\right)\overset{K\to\infty}{\to}\text{Poisson}(\lambda)$, then we get that for each row i, the number of new non-zero columns with a one in the i^{th} row is distributed

$$\text{Poisson}\left(\alpha\left(\psi\left(\sum t+t_i+1\right)-\psi\left(\sum t+1\right)\right)\right). \quad (6)$$

Putting this all together, instead of sampling z_{ik} for each of the infinitely many all-zero columns individually, we sample K_i^{new}, the number of these columns which will have $z_{ik}=1$. The distribution of K_i^{new} is

$$p(K_i^{\text{new}}|X,Z,\alpha)\propto P(X|Z_{\text{new}})P(K_i^{\text{new}}|\alpha), \quad (7)$$

where $P(K_i^{\text{new}}|\alpha)$ is given by Equation (6). To compute $P(X|Z_{\text{new}})$, we must augment Z with K_i^{new} new columns that have a one in only the i^{th} row; this yields Z_{new}. Though K_i^{new} can be arbitrarily large, (7) decays rapidly as K_i^{new} grows, so we can truncate our evaluation at a finite number of columns and sample K_i^{new} from the corresponding multinomial.

3.4 SAMPLING π_k FOR NEW COLUMNS

For each of the new columns generated in the previous step, we must sample an initial value of π_k. Using the same notation as before for edge lengths, if we are sampling π_k for a new column in which only the i^{th} element is non-zero, then we are sampling from

$$\begin{aligned} p(\pi_k|z_k) &= p(\pi_k|z_{(-i)k}=0,z_{ik}=1) \\ &\propto \left(1-(1-\pi_k)^{t_i}\right)(1-\pi_k)^{\sum t}\pi_k^{-1}. \quad (8) \end{aligned}$$

To obtain a sample from this distribution, we use the Metropolis-Hastings sampler from Section 3.2 where we replace equation (4) with (8).

3.5 SAMPLING α

Following Görür et al. (2006), we place a gamma prior, $\mathcal{G}(1,1)$, on α. Noting that α only influences Z through the number of non-zero columns K_i^+, we compute

$$\begin{aligned} p(\alpha|Z) &\propto p(Z|\alpha)p(\alpha) \\ &\sim \text{Poisson}\left(K^+; \alpha\big(\psi\big(1+\sum t\big)-\psi(1)\big)\right)\cdot \mathcal{G}(1,1) \\ &\sim \mathcal{G}\left(K^+ + 1, \psi\Big(1+\sum t\Big) - \psi(1) + 1\right), \end{aligned}$$

where $\sum t$ is the total edge length in the tree.

4 AN APPLICATION TO CHOICE

Choice models play important roles in both econometrics (McFadden, 2000) and cognitive psychology (Luce, 1959). They describe what happens when people are given two or more options and are asked to choose one of them. In this section, we will restrict our attention to choices between pairs of objects, though the methods presented here can be applied more generally.

Even in the simple case of binary decisions, people's choices are not deterministic. The Elimination By Aspects (EBA) model is a popular attempt to explain this variation (Tversky, 1972). EBA hypothesizes that choices are based on a weighted combination of the features of objects. Keeping our earlier notation, let Z be a feature matrix where $z_{ik} = 1$ if the i^{th} object has the k^{th} feature and $z_{ik} = 0$ otherwise. For each of the features, there is a corresponding weight w_k. The higher the weight, the more influence that feature has. The EBA model defines the probability of choosing object i over object j as

$$p_{ij} = \frac{\sum_k w_k z_{ik}(1-z_{jk})}{\sum_k w_k z_{ik}(1-z_{jk}) + \sum_k w_k (1-z_{ik}) z_{jk}}. \quad (9)$$

For comparison with previous results (Görür et al., 2006) we assume extra noise in people's choices, with $\tilde{p}_{ij} = (1-\epsilon)p_{ij} + 0.5\epsilon$.

If X is the observed choice matrix where x_{ij} contains how many times object i was chosen over object j, then for any given w and Z, the probability of X is

$$P(X|Z,w) = \prod_{i=1}^{N}\prod_{i<j}\binom{x_{ij}+x_{ji}}{x_{ij}}\tilde{p}_{ij}^{x_{ij}}(1-\tilde{p}_{ij})^{x_{ji}}. \quad (10)$$

If the number of features is known, Wickelmaier and Schmid (2004) showed how to estimate the weight vector and feature matrix. In general, though, the number of features is not known. Therefore, Görür et al. (2006) applied the IBP to this model in order to simultaneously infer the number of features, the feature matrix, and the weights of these features, and obtained improved performance over previous models.

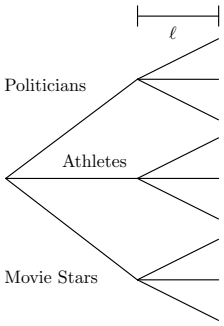


Figure 2: Hypothesis about a tree on preferences (Rumelhart and Greeno, 1971)

In an influential paper, Tversky and Sattath (1979) introduced the preference tree model as an extension of EBA. This model is applicable if the relationships of objects can be captured in a tree structure. In preference trees, each feature has to strictly obey the tree structure. That is, if two objects share a common feature, then all descendents of their most recent common ancestor must have that feature. In some situations, this tree structure may either be known in advance or a good working hypothesis may be available. An example can be found in an experiment reported by Rumelhart and Greeno (1971), in which subjects made 36 pairwise choices of who among a group of nine "well-known personalities" they would like to spend time with. The nine personalities consisted of three politicians, three athletes and three movie stars. It was therefore hypothesized that the tree structure summarizing the prior beliefs about these personalities was similar to the tree shown in Figure 2. In this figure, ℓ is the length of the edge from each general category of people to each individual at the leaf.

Just as the IBP can be used to infer features for EBA, the pIBP defines an appropriate prior for the case where features are assumed to follow a tree structure, as in preference trees. The pIBP model for feature generation can be seen as a soft version of preference trees, allowing features to break the tree structure but assigning low probability to these events. Görür et al. (2006) performed a comparison between the IBP as a prior and EBA models with fixed numbers of features as well as a finite preference tree model that was able to use the tree structure. It was shown that EBA with an IBP prior on the feature matrix outperformed all others. As we will show, using the pIBP gives both quantitatively and qualitatively better results in the case where the features are drawn using a tree.

To complete the full specification of the EBA model, we assume that each object has a unique feature as well as an unknown additional number of features that



Figure 3: Example data demonstrating block structure. (a) True underlying features with corresponding weights in the top row. (b) Underlying probability choice matrix derived from (a) where the lighter the (i, j) entry, the more likely i is to be chosen over j. (c) An example observed choice matrix X drawn from (b) with 5 observations per pair.

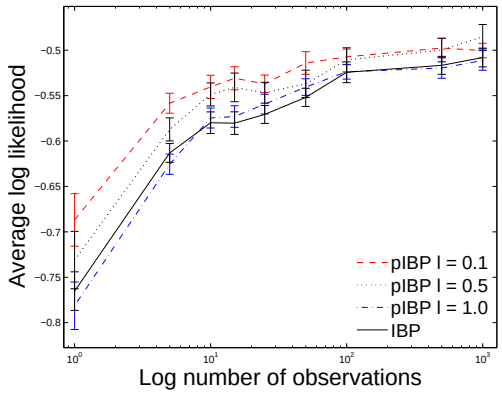


Figure 4: Comparison of the average predictive log-likelihood of the IBP and pIBP models with different degrees of prior knowledge along with error bars. On the x-axis, we vary the (log) number of observations seen for each (i, j) pair.

may or may not be shared as shown in Figure 3(a). We place a pIBP prior on the nonparametric part of the feature matrix and an independent $\mathcal{G}(1, 1)$ prior on each w_k. Inference proceeds as outlined above, with the addition of a Metropolis-Hastings step on w_k. A method similar to that in Görür et al. (2006) was used to deal with the w_k values of new columns.

We generated data from this choice model using the tree from Figure 2 with $\ell = 0.1$. The tree induces a "block structure" in the choice matrix, with the correlated features of objects along each branch resulting in similar patterns of choice for those objects. An example feature matrix generated from this model is shown in Figure 3(a). The top row shows the feature weights of the corresponding columns; the whiter the feature, the more weight that feature has. The feature matrix, Z, is displayed below where entries that are one are white and zero entries are black. Fifteen such examples were generated. For each of these examples, we computed the true value of choosing object i over j as shown in Figure 3(b) where the whiter the $(i, j)^{\text{th}}$ entry, the more likely i is to be chosen over j. Based on these values, we generated data sets with 1, 5, 10, 15, 25, 50, 100, 500, and 1000 choice observations per pair (i, j) following (9). An example of an observed matrix X can be seen in Figure 3(c) with only 5 observations per pair. The lighter the $(i, j)^{\text{th}}$ entry, the more times i was picked over j.

We used these data to examine the effects of two different factors. First, we wished to show the effect of using the pIBP prior over using the IBP prior as the number of observed choice decisions varied. The use of prior knowledge should always help, but with more observations, the influence of the prior should decrease. Second, we wished to test the effect of varying ℓ in our prior. The three values we tested were $\ell = 0.1$, $\ell = 0.5$, and $\ell = 1.0$. As mentioned in Section 2.3, the pIBP with $\ell = 1.0$ is the same as the IBP, but does not integrate over π_k analytically.

For each of the nine observation levels on each of the fifteen examples, we performed leave-one-out cross validation with each model. For each model and validation point at each observation level, we ran an MCMC sampler for 3000 iterations from three different random initialization points. The first 1000 samples from each run were discarded as a burn-in period even though all chains appeared to have mixed within 100 iterations. The predictive likelihood was then averaged across every 10^{th} sample for all configurations. These results can be seen in Figure 4.

As expected, since the pIBP with $\ell = 0.1$ was using the true tree that generated the data, it was able to beat all other configurations for all numbers of observations except 1000, in which case all algorithms performed similarly. As the number of observations increases, the effect of the prior decreases and the models perform more similarly. The pIBP with $\ell = 0.5$ performs better than the IBP, but not as well as the pIBP with the correct tree. This shows that even without perfect knowledge of the tree structure, by inserting some information into the prior, we are able to outperform algorithms that cannot use the same prior knowledge. We also see that the IBP and pIBP with $\ell = 1.0$ perform nearly identically, so explicitly sampling π_k in the inference algorithm does not influence the results.

In addition to obtaining higher likelihoods, the results from the pIBP were also more concise. In Figure 5(a), we show the average feature matrices for the pIBP and IBP when presented with the choice matrix in Figure 3(c). In order to obtain these averages, the Z matrices for all samples after the burn-in period were collapsed into equivalent Z matrices in which the weights of all identical columns were summed together. This results in the same probabilities under the EBA model and allows us to average these values across all samples.

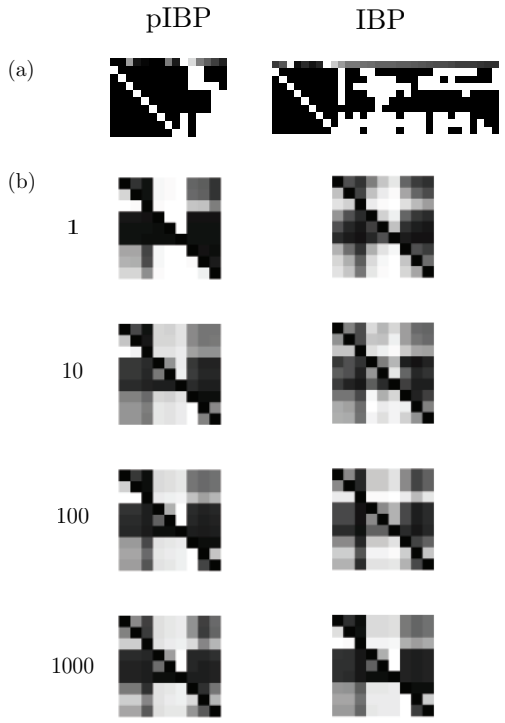


Figure 5: (a) Mean posterior Z matrices with corresponding weights when presented with data from Figure 3. (b) Mean estimated choice matrices P_{ij} for the pIBP and IBP when presented with the different numbers of observations from the data in Figure 3.

We also dropped all columns whose weight was below 0.1. As can be seen, the pIBP recovers a feature matrix very similar to the true data while the IBP requires many more features and still does not achieve the same performance. This large number of features necessary to explain the same data was also observed by Görür et al. (2006). Finally, we checked to see how many examples are needed for the choice probability matrix estimated from the samples of Z and W to show the same structure as the true choice probability matrix from Figure 3(b). In Figure 5(b), we show estimated choice matrices for 1, 10, 100, and 1000 observations per pair. With the pIBP, we observe a block structure immediately, though not all details of the choice matrix are correct. Within very few observations, though, the choice probabilities get close to the true values. In the IBP, we need more than 100 samples before it recovers the block structure.

5 CONCLUSION

We have described the Phylogenetic Indian Buffet Process, a novel non-exchangeable prior for infinite latent feature models. If we can summarize our prior knowledge about the relationships of objects using a tree structure, the pIBP allows us to perform nonparametric Bayesian inference efficiently. This allows us to incorporate our prior knowledge while still harnessing the power of nonparametric Bayesian methods to simultaneously infer both the number of features and their parameters. We have shown through an application to choice models that this can improve the performance of existing methods.

Acknowledgements

This work was supported by the UC Berkeley Chancellor's Faculty Partnership Fund, by grant BCS-0631518 from the National Science Foundation and by a grant from the Lawrence Livermore National Laboratory.

References

P. Diaconis. Recent progress in de Finetti's notions of exchangeability. *Bayesian Statistics*, 3:111–125, 1988.

D. Görür, F. Jäkel, and C. E. Rasmussen. A choice model with infinitely many latent features. In *Proceedings of the 23rd International Conference on Machine learning (ICML)*, 2006.

T. L. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian Buffet Process. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

R. D. Luce. *Individual Choice Behavior*. Wiley, 1959.

D. McFadden. Economic choice. In *Nobel Lectures in Economic Sciences 1996-2000*, 2000.

R. M. Neal. Defining priors for distributions using Dirichlet diffusion trees. Technical Report 0104, Department of Statistics, University of Toronto, 2001.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

D. L. Rumelhart and J. G. Greeno. Similarity between stimuli: An experimental test of the Luce and Restle choice models. *Journal of Mathematical Psychology*, 8 (3):370–381, 1971.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

Y. W. Teh, H. Daumé III, and D. M. Roy. Bayesian agglomerative clustering with coalescents. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

R. Thibaux and M. I. Jordan. Hierarchical beta processes and the Indian Buffet Process. In *Proceedings of the Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

A. Tversky. Elimination by aspects: A theory of choice. *Psychological Review*, 79:281–299, 1972.

A. Tversky and S. Sattath. Preference trees. *Psychological Review*, 86(6):542–573, 1979.

F. Wickelmaier and C. Schmid. A Matlab function to estimate choice model parameters from pair comparison data. *Behavior Research Methods, Instruments, and Computers*, 36:29–40, 2004.

Topic Models Conditioned on Arbitrary Features with Dirichlet-multinomial Regression

David Mimno
Computer Science Dept.
University of Massachusetts, Amherst
Amherst, MA 01003

Andrew McCallum
Computer Science Dept.
University of Massachusetts, Amherst
Amherst, MA 01003

Abstract

Although fully generative models have been successfully used to model the contents of text documents, they are often awkward to apply to combinations of text data and document metadata. In this paper we propose a Dirichlet-multinomial regression (DMR) topic model that includes a log-linear prior on document-topic distributions that is a function of observed features of the document, such as author, publication venue, references, and dates. We show that by selecting appropriate features, DMR topic models can meet or exceed the performance of several previously published topic models designed for specific data.

1 Introduction

Bayesian multinomial mixture models such as latent Dirichlet allocation (LDA) [3] have become a popular method in text analysis due to their simplicity, usefulness in reducing the dimensionality of the data, and ability to produce interpretable and semantically coherent topics. Text data is generally accompanied by metadata, such as authors, publication venues, and dates. Many extensions have been proposed to the basic mixture-of-multinomials topic model to take this data into account. Accounting for such side information results in better topics and the ability to discover associations and patterns, such as learning a topical profile for a given author, or plotting a timeline of the rise and fall of a topic. Currently, developing models for new types of metadata involves specifying a valid generative model and implementing an inference algorithm for that model. In this paper, we propose a new family of topic models based on Dirichlet-multinomial regression (DMR). Rather than generating metadata or estimating topical densities for metadata elements, DMR topic models condition on observed data. As with other conditional models such as maximum entropy classifiers and conditional random fields, users with limited statistical and coding knowledge can quickly specify arbitrarily complicated document features while retaining tractable inference.

The simplest method of incorporating metadata in generative topic models is to generate both the words and the metadata simultaneously given hidden topic variables. In this type of model, each topic has a distribution over words as in the standard model, as well as a distribution over metadata values. Examples of such "downstream" models include the authorship model of Erosheva, Fienberg and Lafferty [5], the Topics over Time (TOT) model of Wang and McCallum [15], the Group-Topic model of Wang, Mohanty and McCallum [16], the CorrLDA model of Blei and Jordan [1] and the named entity models of Newman, Chemudugunta and Smyth [12].

One of the most flexible members of this family is the supervised latent Dirichlet allocation (sLDA) model of Blei and McAuliffe [2]. sLDA generates metadata such as reviewer ratings by learning the parameters of a generalized linear model (GLM) with an appropriate link function and exponential family dispersion function, which are specified by the modeler, for each type of metadata. We show in Section 4.3 that the TOT model is an example of sLDA.

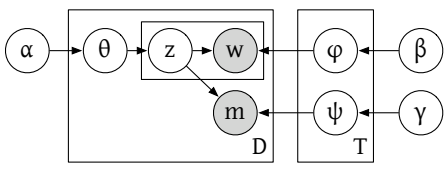


Figure 1: Graphical model representation of a "downstream" topic model, in which metadata m is generated conditioned on the topic assignment variables $\boldsymbol{z}$ of the document and each topic has some parametric distribution over metadata values.

Another approach involves conditioning on metadata elements such as authors by representing document-topic distributions as mixtures of element-specific distributions. One example of this type of model is the author-topic model of Rosen-Zvi, Griffiths, Steyvers and Smyth [13]. In this model, words are generated by first selecting an author uniformly from an observed author list and then selecting a topic from a distribution over topics that is specific to that author. Given a topic, words are selected as before. This model assumes that each word is generated by one and only one author. Similar models, in which a hidden variable selects one of several multinomials over topics, are presented by Mimno and McCallum [11] for modeling expertise by multiple topical mixtures associated with each individual author, by McCallum, Corrada-Emmanuel, and Wang [9] for authors and recipients of email, and by Dietz, Bickel and Scheffer [4] for inferring the influence of individual references on citing papers. These "upstream" models essentially learn an assignment of the words in each document to one of a set of entities.

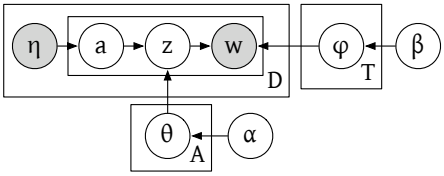


Figure 2: An example of an "upstream" topic model (Author-Topic). The observed authors determine a uniform distribution η over authors. Each word is generated by selecting an author, a, then selecting a topic from that author's topic distribution θ_a, and finally selecting a word from that topic's word distribution.

Previous work in metadata-rich topic modeling has focused either on specially constructed models that cannot accommodate combinations of modalities of data beyond their original intention, or more complicated models such as exponential family harmoniums and sLDA, whose flexibility comes at the cost of increasingly intractable inference. In contrast to previous methods, Dirichlet-multinomial regression (DMR) topic models are able to incorporate arbitrary types of observed continuous, discrete and categorical features with no additional coding, yet inference remains relatively simple.

In section 4 we compare several topic models designed for specific types of metadata to DMR models conditioned on features that emulate those models. We show that performance of DMR models is in almost all cases comparable to similar generative models, and can be considerably better.

2 Modeling the influence of document metadata with Dirichlet-multinomial regression

For each document d, let $\boldsymbol{x}_d$ be a vector containing feature that encode metadata values. For example, if the observed features are indicators for the presence of authors, then $\boldsymbol{x}_d$ would include a 1 in the positions for each author listed on document d, and a 0 otherwise. In addition, to account for the mean value of each topic, we include an intercept term or "default feature" that is always equal to 1.

For each topic t, we also have a vector $\boldsymbol{\lambda}_t$, with length the number of features. Given a feature matrix X, the generative process is:

1. For each topic t,
   (a) Draw $\boldsymbol{\lambda}_t \sim \mathcal{N}(0, \sigma^2 I)$
   (b) Draw $\boldsymbol{\phi}_t \sim \mathcal{D}(\beta)$
2. For each document d,
   (a) For each topic t let $\alpha_{dt} = \exp(\boldsymbol{x}_d^T \boldsymbol{\lambda}_t)$.
   (b) Draw $\boldsymbol{\theta}_d \sim \mathcal{D}(\boldsymbol{\alpha}_d)$.
   (c) For each word i,
      i. Draw $z_i \sim \mathcal{M}(\boldsymbol{\theta}_d)$.
      ii. Draw $w_i \sim \mathcal{M}(\boldsymbol{\phi}_{z_i})$.

The model therefore includes three fixed parameters: σ^2, the variance of the prior on parameter values; β, the Dirichlet prior on the topic-word distributions; and $|T|$, the number of topics.

Integrating over the multinomials $\boldsymbol{\theta}$, we can construct the complete log likelihood for the portion of the model involving the topics $\boldsymbol{z}$:

$$P(\boldsymbol{z}, \boldsymbol{\lambda}) = \prod_d \frac{\Gamma(\sum_t \exp(\boldsymbol{x}_d^T \boldsymbol{\lambda}_t))}{\Gamma(\sum_t \exp(\boldsymbol{x}_d^T \boldsymbol{\lambda}_t) + n_d)} \prod_t \frac{\Gamma(\exp(\boldsymbol{x}_d^T \boldsymbol{\lambda}_t) + n_{t|d})}{\Gamma(\exp(\boldsymbol{x}_d^T \boldsymbol{\lambda}_t))} \times \prod_{t,k} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\lambda_{tk}^2}{2\sigma^2}\right). \quad (1)$$

The derivative of the log of Equation 1 with respect to the parameter λ_{tk} for a given topic t and feature k is

$$\frac{\partial \ell}{\partial \lambda_{tk}} = \sum_d x_{dk} \exp(\boldsymbol{x}_d^T \boldsymbol{\lambda}_t) \times \Big(\Psi\big(\sum_t \exp(\boldsymbol{x}_d^T \boldsymbol{\lambda}_t)\big) - \Psi\big(\sum_t \exp(\boldsymbol{x}_d^T \boldsymbol{\lambda}_t) + n_d\big) + \Psi\big(\exp(\boldsymbol{x}_d^T \boldsymbol{\lambda}_t) + n_{t|d}\big) - \Psi\big(\exp(\boldsymbol{x}_d^T \boldsymbol{\lambda}_t)\big)\Big) - \frac{\lambda_{tk}}{\sigma^2}. \quad (2)$$

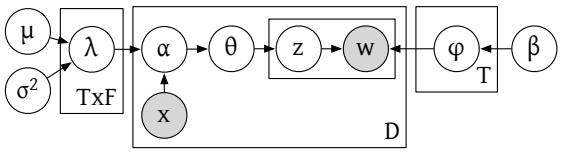


Figure 3: The Dirichlet-multinomial Regression (DMR) topic model. Unlike all previous models, the prior distribution over topics, α, is a function of observed document features, and is therefore specific to each distinct combination of metadata feature values.

We train this model using a stochastic EM sampling scheme, in which we alternate between sampling topic assignments from the current prior distribution conditioned on the observed words and features, and numerically optimizing the parameters $\boldsymbol{\lambda}$ given the topic assignments. Our implementation is based on the standard L-BFGS optimizer [8] and Gibbs sampling-based LDA trainer in the Mallet toolkit [10].

3 Related Work

Recent work by Blei and McAuliffe [2] on supervised topic models (sLDA) combines a topic model with a log-linear (GLM) model, but in the opposite manner: rather than conditioning on observed features through a log-linear model and then predicting topic variables, sLDA uses topic variables as inputs to the log-linear model to generate observed features. An important advantage of the DMR topic model over sLDA for many applications is that DMR is fully conditional with respect to the observed features. In contrast, sLDA must explicitly estimate probability distributions over all possible feature values by fully specifying the link and dispersion functions for a GLM. Although the class of exponential dispersion families supports a wide range of modalities, the specification of GLMs adds modeling complexity. In addition, adding these distributions to the complete log likelihood of the model may result in a significantly more complicated model that is correspondingly more difficult to train.

In contrast, "off the shelf" DMR topic models can be applied to any set of features with no additional model specification. Furthermore, training a model with complex, multimodal, dependent features is no more difficult in a DMR framework than training a model with a single observed real-valued feature. The distinction between conditional and generative methods is analogous to the difference between maximum entropy and naïve Bayes classifiers and between conditional random fields and hidden Markov models.

Guimaraes and Lindrooth [6] use Dirichlet-multinomial regression in economics applications, but do not use a mixture model or any hidden variables. They observe that Dirichlet-multinomial regression falls within the family of overdispersed generalized linear models (OGLMs), and is equivalent to logistic regression in which the output distribution exhibits extra-multinomial variance. This property is useful because DMR produces unnormalized Dirichlet parameters rather than normalized multinomial parameters. These Dirichlet parameters can then be used as a prior for a Bayesian mixture model.

4 Experimental Results

The DMR topic model comprises a broad space of conditional topic models, offering great flexibility for users to define new features. For example, Table 4 shows results for a model incorporating years, venues, authors, and references. In this example, changing two of three authors substantially affects the topical Dirichlet prior. In order to establish that DMR topic models can be effectively trained using the methods described in this paper, we present three examples of DMR models, in which the features are designed to emulate previously published topic models designed for specific types of side information. The purpose of these comparisons is not to suggest that the models compared should necessarily be replaced by equivalent DMR models, but rather to explore the benefits of building custom models relative to simply defining features and passing them to the DMR trainer.

Table 1: DMR topic prior for two documents, given features **2003**, **JMLR**, **D. Blei**, **A. Ng**, and **M. I. Jordan** and features **2004**, **JMLR**, **M.I. Jordan**, **F. Bach**, and **K. Fukumizu**.

α_t	Topic words (Blei, Ng, Jordan)
2.098	models model gaussian mixture generative
0.930	bayesian inference networks network probabilistic
0.692	classifier classifiers bayes classification naive
0.636	probabilistic random conditional probabilities fields
0.614	sampling sample monte carlo chain samples
α_t	Topic words (Jordan, Bach, Fukumizu)
4.046	kernel density kernels data parametric
2.061	space dimensional high reduction spaces
1.780	learning machine learn learned reinforcement
1.501	prediction regression bayes predictions naive
0.879	problem problems solving solution solutions

We evaluate the DMR topic model on a corpus of research papers drawn from the Rexa database.[1] For each paper we have text, a publication year, a publication venue, automatically disambiguated author IDs, and automatically disambiguated references. We se-

[1] http://www.rexa.info

lect a subset of papers from the corpus from venues related to artificial intelligence. We filter out dates earlier than 1987, authors that appear on fewer than five papers, and references to papers with fewer than 10 citations. In addition, for each type of metadata (authors, references, and dates) we train the relevant model only on documents that have that information, since the generative semantics of the Author-Topic model, for example, is undefined if there are no observed authors.

In order to provide a fair comparison and reduce the effect of arbitrary smoothing parameters, we optimize the α_t parameters of all non-DMR topic models using stochastic EM as described by Wallach [14]. This parameter determines the expected mean proportion of each topic. Optimizing the α_t parameters has a substantial positive effect on both model likelihood and held-out performance. Results without hyperparameter optimization are not shown. The DMR model intrinsically represents the mean level of each topic through the parameters for the default feature. The smoothing parameter for the topic-word distributions, β, is constant for all models at 0.01. The variance σ^2 for DMR is set to 10.0 for the default features and 0.5 for all other features. All models are run with 100 topics.

We train each model for 1000 iterations. After an initial burn-in period of 250 iterations we optimize parameters (λ for DMR, α for all other models) every 50 iterations. All evaluations are run over 10-fold cross validation with five random initializations for each fold.

4.1 Author features

For author features, we compare the Author-Topic (AT) model [13] to DMR trained on author indicator features. Example topics for three authors are shown in table 2.

4.1.1 Held-out Likelihood

To evaluate the generalization capability of the model we use the perplexity score described by Rosen-Zvi et al. [13] as well as the empirical likelihood (EL) method advocated for topic model evaluation by Li and McCallum [7]. Evaluating the probability of held-out documents in topic models is difficult because there are an exponential number of possible topic assignments for the words. Both metrics solve this problem by sampling topic distributions from the trained model. Given a trained model, calculating the perplexity score involves sampling topics for half the words in a testing document conditioned on those words. We then use that sampled distribution to calculate the log probability of the remaining words. In the empirical likelihood method, we sample a large number of topic multinomials for each testing document, according to the generative process of the model. We then calculate the log of the average probability of the words given those sampled topic distributions.

Both metrics measure a combination of how good the topic-word distributions are and how well the model can guess which combinations of topics will appear in a given document. The difference is that empirical likelihood estimates the probability of the words without knowing anything about the content of the document, while perplexity also measures the model's ability to "orient" itself quickly given a small amount of local information, such as the first half of the document.

For the EL DMR topic model, we sample $|S|$ unconditional word distributions for a given held-out document d by first calculating the α_d parameters of the Dirichlet prior over topics specific to that document given the observed features $\boldsymbol{x}_d$ in the manner described earlier. We then sample a topic distribution $\boldsymbol{\theta}_{ds}$ from that Dirichlet distribution. Finally, we calculate the probability of each of the observed word tokens w_i by calculating the marginal probability over each topic t of that type using the current point estimates of $P(w_i|t)$ given the topic-word counts.

$$EL(d) \quad = \quad \frac{1}{|S|} \sum_s \sum_i \sum_t \theta_{dts} \frac{n_{w_i|t} + \beta}{n_t + |T|\beta} \qquad (3)$$

Results for perplexity and empirical likelihood for AT and DMR with x_d = author indicator functions are shown in Figure 4. DMR shows much better perplexity than either LDA or AT, while both author-aware models do substantially better than LDA in empirical likelihood. The AT models are consistently slightly better in EL than DMR, but the difference is much less than the difference between DMR and LDA. One explanation for the improved perplexity is that DMR uses a "fresh" Dirichlet prior for each held-out document, which can rapidly adapt to local word information in the test documents. In contrast, AT uses multinomials to represent author-topic distributions. These multinomials have less ability to adapt to the test document, as they generally consist of hundreds of previously assigned words.

4.1.2 Predicting Authors

In addition to predicting the words given the authors, we also evaluate the ability of the Author-Topic (AT) and DMR models to predict the authors of a held-out document conditioned on the words. For each model we can define a non-author-specific Dirichlet prior on topics. For AT, defining a prior over topics is equiv-

Table 2: Ranked topics for three authors under the DMR topic model (left) and the Author-Topic (AT) model (right). For DMR, the sampling distribution for the first word in a document given an author is proportional to the number on the left. For a given topic t, this value is $\exp(\lambda_{t0} + \lambda_{ta} x_{da})$, where λ_{t0} is the default parameter for topic t. For AT, the sampling distribution for the next word $(i + 1)$ in a document given the author is proportional to the number on the left. The integer portion generally corresponds to the number of words in a given topic currently assigned to the author, while the fractional part corresponds to α_t. These values are much larger than those for DMR, meaning that the topic drawn for word $i + 1$ will have relatively little influence on the topic drawn for word $i + 2$.

DMR		**AT**	
	David Blei		**David Blei**
0.25	data mining sets large applications	48.21	bayesian data distribution gaussian mixture
0.24	text documents document categorization large	36.17	text documents document information
0.16	problem work set general information	31.39	model models probabilistic modeling show
0.16	method methods results proposed set	28.09	inference approximation propagation approximate
0.15	distribution bayesian model gaussian models	15.10	markov hidden models variables random
0.15	semantic syntactic lexical sentence named	11.05	discourse sentences aspect semantic coherence
0.14	retrieval information document documents relevance	9.31	process approaches methods techniques terms
0.13	model models show parameters order	9.20	probability distribution distributions estimates
0.13	image images resolution pixels registration	9.11	segmentation image texture grouping region
0.11	translation language word machine english	8.25	data sets set large number
0.11	control robot robots manipulators design	7.28	method methods propose proposed applied
0.10	reasoning logic default semantics theories	6.15	networks bayesian probabilistic inference network
0.10	simple information form show results	5.28	problem problems solving solution solutions
0.09	system systems hybrid intelligent expert	5.19	task tasks performed goal perform
	Andrew Ng		**Andrew Ng**
0.31	show algorithms results general problem	202.11	reinforcement policy state markov decision
0.31	number large size small set	112.30	error training data parameters sample
0.21	system systems hybrid intelligent expert	97.18	learning bounds function bound algorithms
0.20	method methods results proposed set	58.33	show results problem simple class
0.19	results quality performance show techniques	57.36	algorithm algorithms efficient problem set
0.18	algorithm algorithms efficient fast show	54.26	optimal time results computing number
0.17	learning reinforcement policy reward state	39.21	bayesian data distribution gaussian mixture
0.16	decision markov processes mdps policy	34.37	learning learn machine learned algorithm
0.15	feature features selection classification extraction	31.37	work recent make previous provide
0.15	results experimental presented experiments proposed	31.25	set general properties show defined
0.14	performance results test experiments good	31.14	classification classifier classifiers accuracy class
0.14	learning training learn learned examples	31.09	inference approximation propagation approximate
0.14	knowledge representation base acquisition bases	30.19	feature features selection classification performance
0.13	problem work set general information	22.39	model models probabilistic modeling show
	Michael Jordan		**Michael Jordan**
0.69	distribution bayesian model gaussian models	58.18	learning bounds function bound algorithms
0.39	algorithm algorithms efficient fast show	57.09	inference approximation propagation approximate
0.38	show algorithms results general problem	33.21	bayesian data distribution gaussian mixture
0.31	problem work set general information	27.39	model models probabilistic modeling show
0.31	models model modeling probabilistic generative	27.20	probability distribution distributions estimates
0.21	performance results test experiments good	27.05	program programs programming automatic
0.21	problem problems solving solution optimization	24.17	entropy maximum criterion criteria optimization
0.19	learning training learn learned examples	22.33	show results problem simple class
0.15	networks bayesian inference network belief	21.25	set general properties show defined
0.14	data mining sets large applications	20.36	algorithm algorithms efficient problem set
0.12	simple information form show results	20.11	kernel support vector machines kernels
0.12	function functions gradient approximation linear	18.14	methods simple domains current incremental
0.12	methods techniques approaches existing work	18.02	genetic evolutionary evolution ga population
0.12	learning machine induction rules rule	17.19	feature features selection classification performance

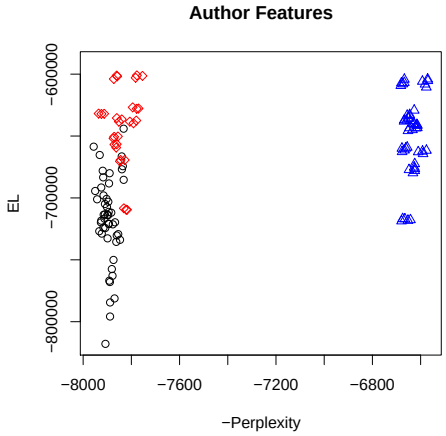


Figure 4: Perplexity and empirical log likelihood for the DMR topic model trained with author indicator features (blue triangles), the Author-Topic (AT) model (red diamonds), and LDA (black circles). Clusters of points represent cross-validation folds. Perplexity is much better for DMR than either AT or LDA. Empirical likelihood is better for the author-aware models than for LDA. In every case, AT performs slightly better in EL than DMR.

alent to adding a single new, previously unseen author for each held-out document. The Dirichlet prior is specified using the α parameters that are fitted in training the model. For DMR, the topic prior Dirichlet is specified using the prior for a document with no observed features (the exponentiated parameters for the intercept terms for each topic).

For each held-out document, we independently sample 100 sequences of topic assignments from the generative process defined by the model, given the word sequence and the topic prior. We add up the number of times each topic occurs over all the samples to get a vector of topic counts $n_1 ... n_{|T|}$. We then rank each possible author by the likelihood function of the author given the overall topic counts. For AT, this likelihood is the probability of adding n_t counts to each author's Dirichlet-multinomial distribution, which is defined by the number of times each topic is assigned to an author $n_{t|a}$ and the total number of tokens assigned to that author n_a:

$$P(d|a) = \frac{\sum_t \alpha_t + n_a}{\sum_t \alpha_t + n_a + \sum_t n_t} \prod_t \frac{\alpha_t + n_{t|a} + n_t}{\alpha_t + n_{t|a}} \quad (4)$$

For DMR, we define a prior over topics given only a particular author as the Dirichlet parameters under the DMR model for a document with only that author feature; in other words, the exponentiated sum of the default feature parameter and the author feature parameter, for each topic. The likelihood for an author is the Dirichlet-multinomial probability of the n_t counts with those parameters. Note that the likelihoods for a given author and held-out document are not necessarily comparable between DMR and AT, but what we are interested in is the ranking.

Results are shown in Figure 5. DMR ranks authors consistently higher than AT.

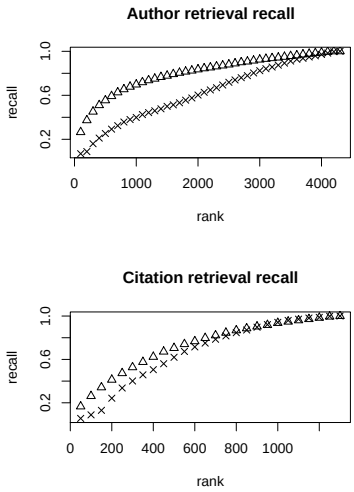


Figure 5: Prediction results for authors and citations. DMR is shown with triangles, and AT and Citation topics with Xes.

4.2 Citation features

Following Dietz, Bickel and Scheffer [4], we consider a model for citation influence that is similar to the Author-Topic model. Each citation is treated as a potential "author", such that when the model generates a word, it first selects a paper from its own references section and then samples a topic from that paper's distribution over topics.

Empirical likelihood results for this citation model are, like AT, slightly better than a DMR model with the same information encoded as citation indicator features. Perplexity was significantly better for the citation model. In this case, the number of occurrences of citations may allow the generative model to obtain a better representation of the topical content of citations. In contrast to authors, each citation's topic multinomial may only consist of a few dozen words, allowing it to adapt more easily to local information.

The DMR model also shows citation prediction performance comparable to the generative citation topic model, as shown in Figure 5.

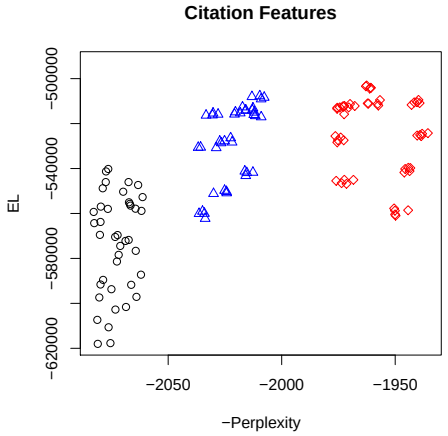


Figure 6: Perplexity and empirical log likelihood for the DMR topic model trained with citation features, the Citation model, and LDA. Unlike other features, citations show very strong perplexity results for the upstream Citation model. DMR continues to outperform LDA in this metric. As with the Author-Topic model, the Citation model has slightly better empirical likelihood than DMR, and both substantially outperform LDA.

4.3 Date features

In many text genres, the date of publication provides information about the content of documents. For example, a research paper published in an artificial intelligence conference in 1997 is much more likely to be about neural networks and genetic algorithms than about support vector machines. The opposite is likely to be true of a paper published in 2005.

Previous work on topic models that take into account time includes the Topics over Time (TOT) model of Wang and McCallum [15]. As with LDA, under the TOT model each word w_i is generated by a hidden topic indicator variable z_i. In addition, the TOT generative process also samples a "date" variable from a topic-specific beta distribution parameterized by ψ_{t1} and ψ_{t2} $\forall t \in T$. The support of the beta distribution is real numbers between zero and one, so rather than generating an actual date, TOT generates a point proportional to the date of a document, within a finite range of dates. We define this proportion, $p_d = \frac{date_d - \min_{d'} date_{d'}}{\max_{d'} date_{d'} - \min_{d'} date_{d'}}$. In order to sample efficiently, Wang and McCallum use the convention that rather than generating the date once per document, each word in a given document generates its own date, all of which happen to be the same.

Consider the terms in the likelihood function for a TOT model that involve p_d for some document d:

$$P(p_d|\boldsymbol{z}_d) = \prod_i \frac{1}{Z_{z_i}} \exp\left(\psi_{z_i 1}\log(p_d) + \psi_{z_i 2}\log(1-p_d)\right) \quad (5)$$

where Z_t is the beta function with parameters ψ_{t1} and ψ_{t2}. Since p_d is constant for every token in a given document, we can rewrite Equation 6 as

$$\frac{1}{Z}\exp\left(\sum_i \psi_{z_i 1}\log(p_d) + \psi_{z_i 2}\log(1-p_d)\right) \quad (6)$$

From this representation, we can see two things. First, this expression is the kernel of a beta distribution with parameters $\sum_i \psi_{z_i 1}$ and $\sum_i \psi_{z_i 2}$, so Z is equal to a beta function with those parameters. Second, this expression defines a generalized linear model. The link function is identity, the exponential dispersion function is beta, and the linear predictor is a function of the number of words assigned to each topic, the topic beta parameters, and the sufficient statistics, which are $\log(p)$ and $\log(1-p)$. With the slight modification of substituting normalized topic counts $\bar{z} = 1/N \sum_i z_i$ for the raw topic counts, we see that TOT is precisely a member of the sLDA family of topic models [2].

To compare DMR regression topic models to TOT, we use the same sufficient statistics used by the beta density: $x_d = \log(p_d)$ and $\log(1-p_d)$. DMR and TOT therefore have the same number of parameters: two for each topic date distribution, plus one parameter (the topic intercept parameter in DMR, an optimized α_t for TOT) to account for the mean proportion of each topic in the corpus.

Figure 7 shows perplexity and EL results for TOT and DMR with TOT-like features. DMR provides substantially better perplexity, while also showing improved empirical likelihood.

5 Conclusions

The Dirichlet-multinomial regression topic model is a powerful method for rapidly developing topic models that can take into account arbitrary features. It can emulate many previously published models, achieving similar or improved performance with little additional statistical modeling or programming work by the user.

One interesting side effect of using the DMR model is efficiency. Adding additional complexity to a topic model generally results in a larger number of variables to sample and a more complicated sampling distribution. Gibbs sampling performance is mainly a function of the efficiency of the innermost loop of the sampler; in the case of LDA this is the calculation of the sampling distribution over topics for a given word. The

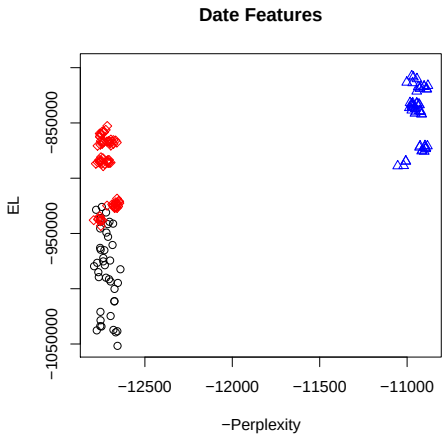


Figure 7: Perplexity and empirical log likelihood for the DMR topic model trained with date features, the Topics Over Time (TOT) model, and LDA. TOT shows perplexity roughly equivalent to LDA, while DMR perplexity is substantially better. DMR also outperforms TOT in empirical likelihood.

Author-Topic model adds an additional set of hidden author assignment variables that must be sampled. TOT adds an additional term (a beta density) to this calculation. In contrast, in a DMR model, all information from the observed document metadata is accounted for in the document-specific Dirichlet parameters. As a result, the sampling phase of DMR training is no more complicated than a simple LDA sampler. The additional overhead of parameter optimization, which we have found decreases as the model converges, can be more than made up by a faster sampling phase, especially if the number of sampling iterations between optimizations is large.

DMR provides a useful complement to generative models such as AT and sLDA, which can make inferences about hidden variables and can be incorporated into more complicated hierarchical models. One area for future work is hybrid sLDA-DMR models constructed by splitting the observed features into a set of conditioned variables and a set of generated variables.

Acknowledgments

Thanks to Xuerui Wang for pointing out the connection between TOT and sLDA, and Hanna Wallach, Rob Hall, and Michael Lavine for helpful discussions.

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant # CNS-0551597, in part by NSF Nanotech # DMI-0531171, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249, and in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under contract number NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

[1] D. Blei and M. Jordan. Modeling annotated data. In *SIGIR*, 2003.

[2] D. Blei and J. D. McAuliffe. Supervised topic models. In *NIPS*, 2007.

[3] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.

[4] L. Dietz, S. Bickel, and T. Scheffer. Unsupervised prediction of citation influences. In *ICML*, 2007.

[5] E. Erosheva, S. Fienberg, and J. Lafferty. Mixed membership models of scientific publications. *PNAS*, 101(Suppl. 1):5220–5227, 2004.

[6] P. Guimaraes and R. Lindrooth. Dirichlet-multinomial regression. Econometrics 0509001, EconWPA, Sept. 2005.

[7] W. Li and A. McCallum. Pachinko allocation: DAG-structured mixture models of topic correlations. In *ICML*, 2006.

[8] D. Liu and J. Nocedal. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3):503–528, 1989.

[9] A. McCallum, A. Corrada-Emmanuel, and X. Wang. Topic and role discovery in social networks. In *IJCAI*, 2005.

[10] A. K. McCallum. MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[11] D. Mimno and A. McCallum. Expertise modeling for matching papers with reviewers. In *KDD*, 2007.

[12] D. Newman, C. Chemudugunta, and P. Smyth. Statistical entity-topic models. In *KDD*, 2006.

[13] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *UAI*, 2004.

[14] H. M. Wallach. Topic modeling: beyond bag-of-words. In *ICML*, 2006.

[15] X. Wang and A. McCallum. Topics over time: a non-Markov continuous-time model of topical trends. In *KDD*, 2006.

[16] X. Wang, N. Mohanty, and A. McCallum. Group and topic discovery from relations and their attributes. In *NIPS*, 2005.

A Polynomial-time Nash Equilibrium Algorithm for Repeated Stochastic Games

Enrique Munoz de Cote*
DEI, Politecnico di Milano
piazza Leonardo da Vinci, 32
20133 Milan, Italy
munoz@elet.polimi.it

Michael L. Littman†
Dept. of Computer Science
Rutgers University
Piscataway, NJ 08854
mlittman@cs.rutgers.edu

Abstract

We present a polynomial-time algorithm that always finds an (approximate) Nash equilibrium for repeated two-player stochastic games. The algorithm exploits the folk theorem to derive a strategy profile that forms an equilibrium by buttressing mutually beneficial behavior with threats, where possible. One component of our algorithm efficiently searches for an approximation of the egalitarian point, the fairest pareto-efficient solution. The paper concludes by applying the algorithm to a set of grid games to illustrate typical solutions the algorithm finds. These solutions compare very favorably to those found by competing algorithms, resulting in strategies with higher social welfare, as well as guaranteed computational efficiency.

1 Problem Statement

Stochastic games (Shapley, 1953) are a popular model of multiagent sequential decision making in the machine-learning community (Littman, 1994; Bowling & Veloso, 2001). In the learning setting, these games are often repeated over multiple rounds to allow learning agents a chance to discover beneficial strategies.

Mathematically, a two-player stochastic game is a tuple $\langle \mathcal{S}, s_0, A_1, A_2, \mathcal{T}, U_1, U_2, \gamma \rangle$; namely, the set of states $\mathcal{S}$, an initial state $s_0 \in \mathcal{S}$, action sets for the two agents A_1 and A_2, with joint action space $\mathcal{A} = A_1 \times A_2$; the state-transition function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$ ($\Pi(\cdot)$ is the set of probability distributions over $\mathcal{S}$); the utility functions for the two agents $U_1, U_2 : \mathcal{S} \times \mathcal{A} \rightarrow \Re$, and the discount $0 \leq \gamma \leq 1$.

In an infinitely repeated stochastic game, the stochastic game is played an unbounded number of rounds. On each round, a stage game is played, starting in s_0 and consisting of a series of state transitions (steps), jointly controlled by the two agents. At each step, both agents simultaneously select their actions, possibly stochastically, via *strategies* π_i (for each agent i). To avoid infinitely long rounds, after each step, the round is allowed to continue with probability γ, otherwise it is terminated. The payoff for a player in a stage game is the total utility obtained before the stage game is terminated. (Note that the continuation probability γ is equivalent to a discount factor.) Players behave so as to maximize their average stage-game payoffs over the infinite number of rounds.

A strategy profile, $\pi = \langle \pi_1, \pi_2 \rangle$, is a *Nash equilibrium* (NE) if each strategy is optimized with respect to the other. In an equilibrium, no agent can do better by changing strategies given that the other agent continues to follow its strategy in the equilibrium. In a repeated game, the construction of equilibrium strategy profiles can involve each player changing strategy from round to round in response to the behavior of the other agent. Note that an ϵ-approximate NE is one in which no agent can do better by more than ϵ by changing strategies given that the other agent continues to follow its strategy in the equilibrium.

Our approach to finding an equilibrium for repeated stochastic games relies on the idea embodied in the *folk theorems* (Osborne & Rubinstein, 1994). The relevant folk theorem states that if an agent's performance is measured via *expected average* payoff, for any *strictly enforceable* (all agents receive a payoff larger than their minimax values) and *feasible* (payoffs can be obtained by adopting some strategy profile) set of average payoffs to the players, there exist equilibrium strategy profiles that achieve these payoffs. The power of this folk theorem is that communally beneficial play, such as mutual cooperation in the Prisoner's Dilemma, can be justified as an equilibrium. A conceptual drawback is

* Supported by The National Council of Science and Technology (CONACyT), Mexico, under grant No. 196839.

†Supported, in part, by NSF IIS-0325281.

that there may exist infinitely many feasible and enforceable payoffs (and therefore a daunting set of equilibrium strategy profiles to choose from). We focus on the search for a special point inside this (possibly infinite) set of solutions that maximizes the minimum advantage obtained by the players. (The *advantage* is the improvement a player gets over the payoff it can guarantee by playing defensively.) We call this point the egalitarian point, after Greenwald and Hall (2003). Other points can also be justified, such as the one that maximizes the product of advantages—the Nash bargaining solution (Nash, 1950).

Earlier work (Littman & Stone, 2005) has shown that the folk theorem can be interpreted computationally, resulting in a polynomial-time algorithm for repeated games. In the prior work, the game in each round is represented in matrix form—each strategy for each player is explicitly enumerated in the input representation. This paper considers the analogous problem when each stage game is represented much more compactly as a stochastic game. Representing such games in matrix form would require an infinitely large matrix since the number steps per round, and therefore the complexity of the strategies, is unbounded. Even if we limit ourselves to stationary deterministic strategies, there are exponentially many to consider.

Concretely, we address the following computational problem. Given a stochastic game, return a strategy profile that is a Nash equilibrium—one whose payoffs match those of the egalitarian point—of the average payoff repeated stochastic game in polynomial time. In fact, because exact Nash equilibria in stochastic games can require unbounded precision, our algorithm returns an arbitrarily accurate approximation.

2 Background

Here, we present background on the problem.

2.1 Minimax Strategies

Minimax strategies guarantee a minimum payoff value, called the security value, that an agent can guarantee itself by playing a *defensive* strategy. In addition, an agent can be held to this level of payoff if the other agent adopts an aggressive *attack* strategy (because minimax equals maximin). Given that minimax strategies guarantee a minimum payoff value, no rational player will agree on any strategy in which it obtains a payoff lower than its security value. The pair of security values in a two-player game is called the *disagreement point*.

The set $X \subseteq \mathbb{R}^2$ of average payoffs achievable by strategy profiles can be visualized as a region in the x-y plane. This region is convex because any two strategy profiles can be mixed by alternating over successive rounds to achieve joint payoffs that are any convex combination of the joint payoffs of the original strategy profiles. The disagreement point $v = (v_1, v_2)$ divides the plane into two regions (see Figure 1): a) the region of mutual advantages (all points in X, above and to the right of v), denotes the strictly enforceable payoff profiles; and b) the relative complement of the region of mutual advantage, which are the payoff profiles that a rational player would reject.

In general-sum bimatrix games, the disagreement point can be computed exactly by solving two zero-sum games (von Neumann & Morgenstern, 1947) to find the attack and defensive strategies and their values. In contrast, the solution to any zero-sum stochastic game can be approximated to any degree of accuracy $\epsilon > 0$ via value iteration (Shapley, 1953). The running time is polynomial in $1/(1-\gamma)$, $1/\epsilon$, and the magnitude of the largest utility $U_{\max}$.

2.2 Markov Decision Processes

In this paper, we use Markov decision processes (Puterman, 1994), or MDPs, as a mathematical framework for modeling the problem of the two players working together as a kind of *meta*-player to maximize a weighted combination of their payoffs. For any weight $[w, 1-w]$ $(0 \leq w \leq 1)$ and point $p = (p_1, p_2)$, define $\sigma_w(p) = wp_1 + (1-w)p_2$.

Note that any strategy profile π for a stochastic game has a value for the two players that can be represented as a point $p^\pi \in X$. To find the strategy profile π for a stochastic game that maximizes $\sigma_w(p^\pi)$, we can solve MDP(w), which is the MDP derived from replacing the utility $r = (r_1, r_2)$ in each state with $\sigma_w(r)$.

2.3 Other Solutions for Stochastic Games

There are several solution concepts that have been considered in the literature. Generally speaking, a Nash equilibrium (NE) is a vector of independent strategies in which all players optimize their independent probability distributions over actions with respect to expected payoff. A correlated equilibrium (CE) allows for dependencies in the agent's randomizations, so a CE is a probability distribution over *joint* spaces of actions. Minimax strategies maximize payoff in the face of their worst opponent. At the other extreme, "friend" strategies maximize behavior assuming the opponents are working to maximize the agent's own utility. Friend strategies are appropriate in purely cooperative settings but can perform very badly in mixed incentive settings.

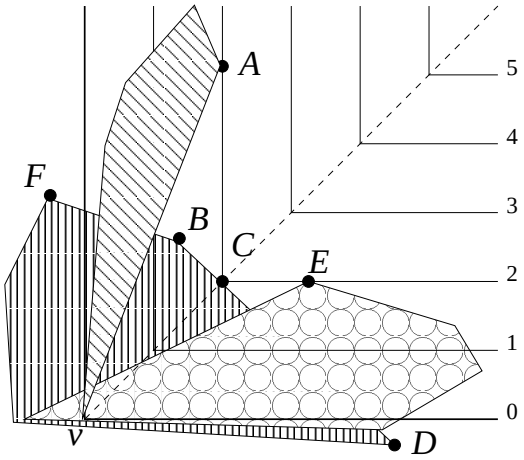


Figure 1: The convex hull $X \subseteq \mathbb{R}^2$ of all payoff profiles. Point $v \in X$ depicts the disagreement point. Three different convex hulls are illustrated. The diagonal line is the egalitarian line.

A powerful and general algorithmic approach for sequential decision problems is value iteration (Bellman, 1957). Variants of value iteration can be defined for each of the solution concepts described above (Zinkevich et al., 2005).

3 Algorithm Description

Of all feasible Nash equilibria, we are interested in one whose payoffs match the egalitarian point, thus maximizing the minimum of the payoffs of the two players. Mathematically, we are searching for a point $P = \text{argmax}_{x \in X} \min_v(x)$. Here, $\min_v(x)$ is the *egalitarian value* of x, meaning $\min(x_1 - v_1, x_2 - v_2)$ where $x = (x_1, x_2)$ and v is the disagreement point. Note that $\min_v(P) \geq 0$ because X is convex and v is the disagreement point—there is a strategy in which both players do at least as well as v.

Define the *egalitarian line* to be the line corresponding to the payoffs in which both player's payoffs are equally high above the disagreement point. Consider the two "friend" solutions to the game, where L is the value to the two players when maximizing Player 2's payoff and R is the value to the two players when maximizing Player 1's payoff. Because X is convex, the egalitarian point P is either L, R, or the (highest) intersection between X and the egalitarian line.

Figure 1 illustrates these three situations with three different example X sets. The solid L-shaped lines in the figure are the contour lines with equal egalitarian values. The egalitarian point in X is the one that reaches the topmost contour line. In the set filled with diagonal lines, the point A is the egalitarian point and $\min_v(A) = 2$. All other points are to the left of A, so A is the point with maximum x coordinate. In the set filled with circles, E is the egalitarian point with $\min_v(E) = 2$. All other points are below E, so E is the point with maximum y coordinate.

The intermediate region with the vertical fill lines is a bit more complex. Point F has the largest y coordinate, but its egalitarian value is negative because of the x coordinate. Point D has the largest x coordinate, but its egalitarian value is negative because of the y coordinate. The point C, which is a linear combination of the vertices B and D, is the egalitarian point, again with a value of 2.

Finding points like E and A is easy—it is just a matter of solving the "friend" MDPs derived using weights of $[0, 1]$ and $[1, 0]$ and halting if either R is on the left or L is on the right of the egalitarian line. However, finding point C is harder, since we need to find points B and D and then intersect the line between them with the egalitarian line.

Figure 2 presents the overall FolkEgal algorithm for finding a strategy profile that achieves the egalitarian point in stochastic games. FolkEgal(U_1, U_2, ϵ) works for utility functions U_1 and U_2 and seeks an equilibrium with accuracy ϵ. The routine $(\delta_i, \alpha_{-i}, v_i) :=$ Game(U_i, ϵ) solves the zero-sum game with utility function U_i to accuracy ϵ. It returns v_i and δ_i, which are the value and strategy (respectively) for the maximizing player i, and α_{-i}, which is the attack strategy of i's opponent $(-i)$. We do not provide code for this subroutine as any zero-sum stochastic game solver can be used. Littman and Stone (2005) provide details on using the attack strategies to stabilize the discovered mutually beneficial payoffs, which we do not repeat here.

The key missing subroutine is $(P, \pi) :=$ EgalSearch(L, R, T), which finds the intersection between the convex hull of payoffs with the egalitarian line. It returns the egalitarian point P and a strategy profile π that achieves it. It is given a point $L \in X$ to the left of the egalitarian line, a point $R \in X$ to the right of the egalitarian line, and a bound T on the number of iterations. Section 4 explains how to choose T.

The algorithm is laid out in Figure 3. Its basic structure is a kind of binary search. On each iteration, it solves an MDP to try to find a policy closer to the egalitarian line. It makes use of several support subroutines. The call $w :=$ Balance(L, R) returns the weight w for which $\sigma_w(L) = \sigma_w(R)$. It can be found by solving a linear equation. The payoff of the optimal strategy profile π for w should be an improvement on L and R with respect to the weight w, that is,

Define FolkEgal(U_1, U_2, ϵ):
  // Find "minimax" strategies
  Let $(\delta_1, \alpha_2, v_1) :=$ Game($U_1, \epsilon/2$)
  Let $(\delta_2, \alpha_1, v_2) :=$ Game($U_2, \epsilon/2$)
  // Make the disagreement point the origin
  Let $v := (v_1, v_2)$
  Let $U_1 := U_1 - v$
  Let $U_2 := U_2 - v$
  // Find "friend" strategies
  Let $(R_0, \pi_2) :=$ MDP($[1, 0]$)
  Let $(L_0, \pi_1) :=$ MDP($[0, 1]$)
  // Find the egalitarian point and its policy
  If R is left of the egalitarian line:
    Let $(P, \pi) := (R_0, \pi_2)$
  Elseif L is to the right of the egalitarian line:
    Let $(P, \pi) := (L_0, \pi_1)$
  Else:
    Let $(P, \pi) :=$ EgalSearch(L_0, R_0, T)
  // If game is like zero sum, compete
  If $\min_v(P) \leq \epsilon$:
    Return (δ_1, δ_2)
  // Else, mutual advantage
  Return π, modified to use threat strategies
    α_1 and α_2 to enforce the equilibrium

Figure 2: Our approach to finding the egalitarian point and a strategy profile that achieves it.

Define EgalSearch(L, R, T):
  If $T = 0$:
    Return Intersect(L, R)
  Let $w :=$ Balance(L, R)
  Let $(P, \pi) :=$ MDP(w)
  If $P \cdot w = L \cdot w$:
    Return Intersect(L, R)
  If P is to the left of egalitarian line:
    Return EgalSearch($P, R, T - 1$)
  Else:
    Return EgalSearch($L, P, T - 1$)

Figure 3: Our search algorithm for intersecting the convex payoff region with the egalitarian line.

$\sigma_w(P^\pi) \geq \sigma_w(R) = \sigma_w(L)$. If it is not strictly better, the search ends. Otherwise, the new point is used as either L or R and it continues.

The final strategy profile returned is found via Intersect(L, R), which discovers the right way to alternate between L and R to produces a payoff on the egalitarian line. Again, a simple linear equation suffices to identify this strategy profile.

Figure 4 illustrates a step of the algorithm. First, note the disagreement point v and the egalitarian line heading out from it. The algorithm is given points L and R such that L is on the left of the egalitarian line and R is on the right. In the diagram, the line passing through L labeled $\bar{L}$ is the set of points p such that $\sigma_{w_L}(p) = \sigma_{w_L}(L)$. Since L was returned as the maximum payoff with respect to some weight w_L, none

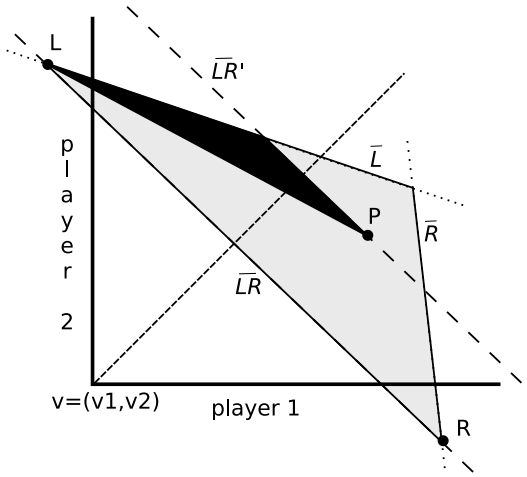


Figure 4: An illustration of the behavior of EgalSearch.

of the points in the convex set can be above this line. Similarly, w_R is the weight that was used in the derivation of R and therefore no payoffs are possible beyond the $\bar{R}$ line in the figure.

Next, notice that both L and R are the payoffs for some strategy profile, so both lie in the convex set. Furthermore, any payoff on the line between L and R can also be achieved by some strategy profile. The weight w, derived by Balance(L, R), is the weight such that every payoff p along the line between L and R has the same weighted value $\sigma_w(p)$. The line is called $\bar{LR}$ in the figure.

Putting these ideas together, consider what happens when we solve MDP(w). We know the result will be at least as high as the $\bar{LR}$ line, since we already know there is a strategy profile that can achieve this payoff. However, we also know that it can't go above the $\bar{R}$ and $\bar{L}$ lines. So, the solution is constrained to lie inside the gray shaded triangle in the figure.

The point P is the hypothetical solution to MDP(w). Since it is on the right of the egalitarian line, it replaces R in the next iteration. The black triangle represents the region that will be searched in the next iteration. In the next section, we show that each iteration reduces the area of this triangle substantially, and thus that a small number of iterations are needed to reduce its intersection with the egalitarian line to $\epsilon/2$.

4 Algorithm Analysis

The main open issue is to set the parameter T, which controls the maximum number of search iterations in EgalSearch. Since solving MDPs and the various other steps each take polynomial time, the overall runtime

of FolkEgal is polynomial if and only if T is bounded by a polynomial.

Let's say we are given a triangle with area ν where the corners are possible joint payoffs. Let point p be the point in the triangle that maximizes $\min_v(p)$. Let point r be the point along the longest edge of the triangle that maximizes $\min_v(r)$.

Claim 1: $\min_v(p) - \min_v(r) \leq \sqrt{2\nu}$.

To see why, let's consider two facts.

1. For points x and y, if $\|x - y\|_2 \leq \delta$, then $y = x + \Delta$ for some $\Delta = (\Delta_1, \Delta_2)$ where $|\Delta_1| \leq \delta$ and $|\Delta_2| \leq \delta$. It follows that $\min_v(y) = \min_v(x+\Delta) \leq \min_v(x + \delta) = \min_v(x) + \delta$. Reversing x and y, we find $|\min_v(x) - \min_v(y)| \leq \delta$.

2. A triangle with longest edge b must have an altitude, h, where $h \leq b$, otherwise it would not fit inside the triangle. Therefore, its area is $\nu = 1/2bh \geq 1/2h^2$. Thus, $h \leq \sqrt{2\nu}$. This argument shows that for any point x in the triangle and y on the largest side, $\|x - y\|_2 \leq h \leq \sqrt{2\nu}$.

Combining these two facts proves the claim: $|\min_v(p) - \min_v(r)| \leq \sqrt{2\nu}$.

Claim 2: Figure 4 shows the generic situation in which the algorithm has found points L and R using weights that result in the edges labeled with $\bar{L}$ and $\bar{R}$. The gray triangle is the remaining region to search. The algorithm then performs an optimization that uncovers a point P inside this region using weight w. The process then repeats with the black triangle. Note:

1. The angle at the "top" of the triangle gets wider each iteration.

   The fact follows because the new top vertex is interior to the main triangle.

2. The area of the black triangle is less than or equal to half of that of gray triangle.

   You can visualize the gray triangle as consisting of three shapes—a top triangle, a trapezoid, and the black triangle. Note that the black triangle and the trapezoid share the same height, but the large base of the trapezoid ($\bar{LR}$ line) is longer than the base of the black triangle on line $\bar{LR}'$ (because the gray triangle tapers). Therefore, the black triangle is smaller than the trapezoid and so is less than half of the area of the gray triangle.

Combining the claims, let p_T be the point that maximizes $\min_v(p_T)$ in the triangle active in the Tth iteration. Let r_T be the point that maximizes $\min_v(r_T)$ on the longest edge of the triangle active in the Tth iteration. Let ν_T be the area of the triangle active in the Tth iteration.

1. $\nu_T \leq \nu_0 \times 1/2^T$
2. $\min_v(p_T) \leq \min_v(r_T) + \sqrt{2\nu_T}$.

So, $\min_v(p_T) \leq \min_v(r_T) + \sqrt{2\nu_0/2^T}$.

If we want the difference to be smaller than, say, ϵ, we need $\sqrt{2\nu_0/2^T} \leq \epsilon$, or $T \geq \log(2\nu_0/\epsilon^2)$.

Since $\nu_0 \leq U_{\max}^2$, the number of iterations is polynomially bounded in the main parameters of the problem and the approximation factor. Each iteration runs in polynomial time.

5 Experimental Results

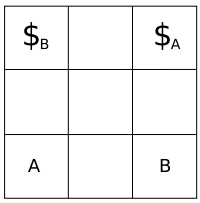
To illustrate the kinds of equilibria produced by our algorithm and to compare them to existing algorithmic approaches, we devised a family of stochastic games played on grids.

All are games played by players A and B. The grids differ in structure, but they all use the same dynamics. Both players A and B occupy distinct cells of the grid and can choose one of 5 distinct actions: N, S, E, W and stand. Actions are executed simultaneously and transitions from one cell to another are deterministic unless a) there is a semi-passable wall in between cells (depicted as a dotted wall in Figure 5(b)), in which case the player transitions to the desired cell with probability 1/2, or, b) both players attempt to step into the same cell, where the collision is resolved at random by a coin flip, so only one player ends up occupying the desired cell and the other makes no transition. Walls are impassable and players cannot pass through each other—attempts to do so result in no transition.

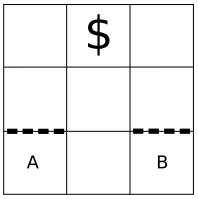
Goal locations can be specific for some agent X (depicted as a dollar sign with subindex, for example, 5(a), 5(c), 5(d), 5(e)) or general (depicted as a dollar sign without subindex, for example Figures 5(b) and 5(c)). The game ends after any player gets to one of its goals and a goal reward is given. There is a step cost for each of the actions N, S, E, W, but stand has no cost. Note that games are general sum in that it is possible for both players to score by both reaching goals simultaneously.

All games use $\gamma = 0.95$, $\$ = \$_A = \$_B = 100$ and step cost $= -1$. One exception is that the step cost $= -10$ for the asymmetric game[1].

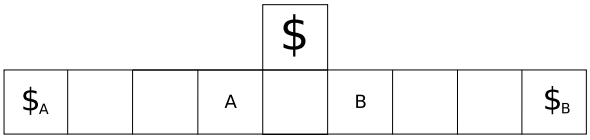
[1]This example can be reconstructed with a step cost of



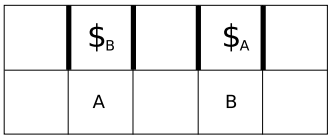
(a) coordination



(b) chicken



(c) prisoner's dilemma



(d) compromise



(e) asymmetric

Figure 5: Grid games in their initial state. $\$_X$ = goal for agent X, \$ = common goal

In our results, we include runs for four solution algorithms. Security-VI uses minimax for each player. It is guaranteed to find an equilibrium in zero-sum games, but not in general. Friend-VI uses a self-regarding optimization strategy for both players—each behaves under the assumption that the other player is trying to help it (Littman, 2001). Such an approach can work well in identical payoff games, but since policies are computed independently, it need not. CE-VI[2] computes a correlated equilibrium for the players at each state. As a result, actions are guaranteed to be coordinated, but such an algorithm need not converge to a Nash equilibrium in general. Our FolkEgal algorithm will always find a Nash equilibrium of the repeated game.

one, but many more states are needed, so we decided to keep the example small by modifying the step cost.

[2]CE-VI stands for all variants (Greenwald & Hall, 2003) of CE (uCE, eCE, rCE), as their results were identical.

We now present results with the set of grid games in Figure 5. For each game, for each solution algorithm, we present the expected payoffs for each agent along with an informal description of the returned strategy profiles.

5.1 Coordination

algorithm	agent 1	agent 2
security-VI	0.0	0.0
friend-VI	45.7	45.7
CE-VI	82.8	82.8
FolkEgal	82.8	82.8

This game is not terribly interesting, but the fact that the players need to pass by each other without colliding makes it relevant to consider their interaction. Friend-VI is unable to coordinate with the opponent and sometimes will collide. Security-VI finds that the worst opponent can always block the goal, so players stay still forever to avoid step costs. Both CE-VI and FolkEgal find a Nash equilibrium by avoiding each other *en route* to goal, and both achieve optimal behavior.

5.2 Chicken

algorithm	agent 1	agent 2
security-VI	43.7	43.7
friend-VI	42.7	42.7
CE-VI	88.3	43.7
FolkEgal	83.6	83.6

This game has an element of the game "chicken" in that both players prefer taking the center path, but given that the other player is taking the center path, the side path is more attractive. We used a variation of standard grid game (Hu & Wellman, 2003) in which collisions are resolved by a coin flip (Littman, 1994) and there is no explicit collision cost.

The difference between security-VI and friend-VI in this game is how an agent behaves if it cannot make it to the center square. In the defensive strategy, once a player does not get the center it will stay put because it assumes (rightly) that the opponent will proceed directly to the goal. Friend-VI, on the other hand, will naively continue moving toward the goal under the assumption that the other player will let it pass. It incurs a small step cost for its overly optimistic outlook.

CE-VI finds an asymmetric solution in which one player is assigned to take the center and the other one uses the side passage, through the semi-passable wall. This policy is a Nash equilibrium in that neither player can improve its reward unilaterally.

FolkEgal finds the solution halfway (0.5) along the edge between vertices $(83.14, 84.05)$ and $(84.05, 83.14)$. These points correspond to strategies in which one

player takes the center and continues beside the goal, waiting for the other player to catch up. The two players reach the goal at the same time. The first player to go through the center incurs a slightly higher cost because it must step around the goal before waiting, hence the asymmetric (but close) values. The weight of .5 means that each strategy is played with equal frequency (strict alternation, say), in the equilibrium.

Note that both players score slightly worse than the dominant player found in the CE-VI solution, due to the cost of coordination. However, both the value obtained by the minimum player *and* the total reward for the two players is better for the FolkEgal algorithm than for CE-VI.

5.3 Prisoner's Dilemma

algorithm	agent 1	agent 2
security-VI	46.5	46.5
friend-VI	46.0	46.0
CE-VI	46.5	46.5
FolkEgal	88.8	88.8

This game was designed to mimic the Prisoner's dilemma. The main choice faced by each of the two agents is whether to move toward the shared goal location in the center or whether to attempt to reach the goal location further out on the side. If both players move toward the center, each has a 50–50 chance of making it to the goal in two steps. If both players move toward the sides, each has a 100% chance of reaching the goal in 3 steps. Clearly, moving to the side is better. However, whichever decision its opponent makes (side or center), the player scores higher by moving to the center.

The results closely match what happens when bimatrix-game versions of the algorithms are applied to Prisoner's dilemma. Security-VI and CE-VI find strategies where both players move to the center (defect). This strategy profile is a Nash equilibrium. Friend-VI is similar, although (as above) once a player is unable to take the center, it continues to try to do so assuming the other player will voluntarily get out of the way. Again, this behavior results in additional unnecessary step costs and is not an equilibrium.

We had expected FolkEgal to find a solution where both players move to their side goals, reserving the center square as a threat (much like tit-for-tat). In fact, FolkEgal found a slightly better scheme—one agent gets the closer common goal (saving step costs) but waits for the other to get to its private goal before entering. FolkEgal find points $(89.3, 88.3)$ and $(88.3, 89.3)$, and players alternate.

5.4 Compromise

algorithm	agent 1	agent 2
security-VI	0.0	0.0
friend-VI	−20.0	−20.0
CE-VI	68.2	70.1
FolkEgal	78.7	78.7

This game is much like the coordination game, but with the twist that it is not possible for a player to reach its goal without the other player stepping aside.

Security-VI adopts the worst-case assumption that the other player will not step aside. Both players end up staying still, as a result, to avoid step costs. Friend-VI is actually even worse. Since both player assumes the other will step aside, the two players simply ram each other indefinitely.

CE-VI converges to a very interesting strategy. Player A steps into Player B's goal and waits. Player A is blocking Player B from scoring, but it is also allowing Player B to pass. Player B walks to the upper left corner and Player A moves back to its initial position. Note that both players are now 3 steps from their respective goals. At this point, both players move to their goals and arrive simultaneously. One of the more interesting aspects of this strategy profile is that Player B waits in the corner until Player A has stepped out of the goal. The reason is that Player A will not attempt to reach its own goal until it is sure it won't be beaten by Player B. Player B, by keeping a respectful distance, signals to Player A that it is safe to move and both players benefit. Since both players are also choosing actions in their own best interest, the resulting strategy profile is an equilibrium.

This strategy profile, while ingenious (and unexpected to us), does not maximize the value of the minimum player. FolkEgal's solution is to alternate between $L = (79.6, 77.7)$ and $R = (77.7, 79.6)$. The strategy profile, in this case, corresponds to one player moving to the space between the goals, the other moving in front of its goal and waiting, then both players reaching their goals together in 5 steps.

5.5 Asymmetric

algorithm	agent 1	agent 2
security-VI	0.0	0.0
friend-VI	−200.0	−200.0
CE-VI	32.1	42.1
FolkEgal	37.2	37.2

This game was designed to show how the algorithms react to an asymmetric starting position. Once again, overly optimistic (friend-VI) and pessimistic (security-VI) assumptions result in very low scores for both players.

Note that the players again need to compromise. Without Player B's cooperation, Player A cannot reach its near goal on the right. It also cannot reach its far goal on the left, because Player B can trail behind it and reach its goal before Player A arrives.

CE-VI discovers that Player B can "offer" Player A a compromise by hanging back exactly one square when Player A moves to the left. As a result, both players reach their goal locations on the left simultaneously. The solution is a Nash equilibrium, although it is not the egalitarian solution.

FolkEgal finds the egalitarian solution as a weighted combination of the points $L = (32.13, 42.13)$ and $R = (85, -10)$ with weight approximately .1. Note that point L corresponds to the solution found by CE-VI. Point R corresponds to the strategy where Player B moves to the right and lets Player A reach the near goal location.

5.6 Summary

There are a few interesting generalizations to make, based on these results. First, although CE-VI is not guaranteed to find a Nash equilibrium, it did so in all 5 games (including games that were designed specifically to thwart it). We were surprised at the robustness of the algorithm.

CE-VI only found the egalitarian solution in one game, however, whereas FolkEgal found it every time. FolkEgal also has guaranteed polynomial runtime bounds, whereas CE-VI is known to fail to converge in some games (Zinkevich et al., 2005). Friend-VI performed uniformly badly and security-VI, or foe-VI (Littman, 2001), often returned a Nash equilibrium, but one with worse overall performance.

6 Future Work

This paper shows how an egalitarian Nash equilibrium solution can be found efficiently for repeated stochastic games. Future work will attempt to generalize these techniques to repeated games on trees (Littman et al., 2006) or DAGs and perhaps even repeated partial information games (Koller et al., 1996).

References

Bellman, R. (1957). *Dynamic programming.* Princeton, NJ: Princeton University Press.

Bowling, M., & Veloso, M. (2001). Rational and convergent learning in stochastic games. *Proceedings of the Seventeenth International Conference On Artificial Intelligence (IJCAI-01)* (pp. 1021–1026). San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Greenwald, A., & Hall, K. (2003). Correlated-Q learning. *Proceedings of the Twentieth International Conference on Machine Learning* (pp. 242–249).

Hu, J., & Wellman, M. P. (2003). Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, *4*, 1039–1069.

Koller, D., Megiddo, N., & von Stengel, B. (1996). Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, *14*, 247–259.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 157–163).

Littman, M. L. (2001). Friend-or-foe Q-learning in general-sum games. *Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 322–328). Morgan Kaufmann.

Littman, M. L., Ravi, N., Talwar, A., & Zinkevich, M. (2006). An efficient optimal-equilibrium algorithm for two-player game trees. *Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI-06).*

Littman, M. L., & Stone, P. (2005). A polynomial-time Nash equilibrium algorithm for repeated games. *Decision Support Systems*, *39*, 55–66.

Nash, J. F. (1950). The bargaining problem. *Econometrica*, *28*, 155–162.

Osborne, M. J., & Rubinstein, A. (1994). *A course in game theory.* The MIT Press.

Puterman, M. L. (1994). *Markov decision processes—discrete stochastic dynamic programming.* New York, NY: John Wiley & Sons, Inc.

Shapley, L. (1953). Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, *39*, 1095–1100.

von Neumann, J., & Morgenstern, O. (1947). *Theory of games and economic behavior.* Princeton, NJ: Princeton University Press.

Zinkevich, M., Greenwald, A. R., & Littman, M. L. (2005). Cyclic equilibria in Markov games. *Advances in Neural Information Processing Systems 18.*

Explanation Trees for Causal Bayesian Networks

Ulf H. Nielsen[1]
uln@zurich.ibm.com

Jean-Philippe Pellet[1,2]
jep@zurich.ibm.com

André Elisseeff[1]
ael@zurich.ibm.com

[1] Business Optimization Group
IBM Research GmbH
8803 Rüschlikon, Switzerland

[2] Machine Learning Group
Swiss Federal Institute of Technology Zurich
8092 Zurich, Switzerland

Abstract

Bayesian networks can be used to extract explanations about the observed state of a subset of variables. In this paper, we explicate the desiderata of an explanation and confront them with the concept of explanation proposed by existing methods. The necessity of taking into account causal approaches when a causal graph is available is discussed. We then introduce causal explanation trees, based on the construction of explanation trees using the measure of causal information flow (Ay and Polani, 2006). This approach is compared to several other methods on known networks.

1 INTRODUCTION

A Bayesian network (BN, Pearl, 1988) is an algebraic tool to compactly represent the joint probability distribution of a set of variables $\mathbf{V}$ by exploiting conditional independence amongst variables. It represents all variables in a directed acyclic graph (DAG), where the absence of arcs between nodes denotes (conditional) independence. In addition to graphically representing the structure of the dependencies between the variables, BNs allow inference tasks to be solved more efficiently. In this paper, we discuss the extraction of explanations in *causal BNs* (Pearl, 2000; Spirtes et al., 2001)—BNs where the arcs depict direct cause–effect relationships between variables.

Generally, explanations in BNs can be classified in three categories (Lacave and Diez, 2002) depending on the focus of the explanation:

- Explanation of *evidence*. Given a subset of observed (instantiated) variables $\mathbf{O} \subsetneq \mathbf{V}$, what is the state of (some of) the other variables $\mathbf{V} \setminus \mathbf{O}$ that best explains $\mathbf{O} = \mathbf{o}$?
- Explanation of the *reasoning process*. When we have received some evidence and belief states are updated by probabilistic inference, how was the reasoning process by which we arrive at this state?
- Explanation of the *model*, which provides insight into the static components of a network such as (conditional) independence relationships, causal mechanisms, etc.

We shall focus our attention on the explanation of evidence: we wish to explain why variables in $\mathbf{O}$ took on specific observed values using assignments in $\mathbf{V} \setminus \mathbf{O}$. To this purpose, we discuss in section 2 the requirements of such an explanation. In section 3, we list the standard approaches to evidence explanation as well as some recent methods to make explanations more concise, and explain some of their drawbacks. We then present causal information trees in section 4, and detail experiments and comparisons in section 5.

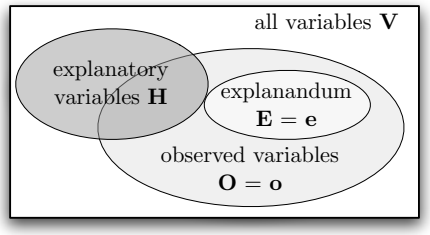
NOTATION

Boldface capitals denote sets of random variables or nodes in a graph, depending on the context. $\mathbf{V}$ is the set of all variables in the analysis. Italicized capitals like X, X_i, Y are random variables or nodes and elements of $\mathbf{V}$; calligraphic capitals such as $\mathcal{X}, \mathcal{Y}$ are their respective domains. Vectors are denoted boldface lowercase, as $\mathbf{e}$ or $\mathbf{p}$; scalars in italics. Unless otherwise stated, the scalars x, y are assumed to be a value of their respective uppercase random variable. The probability distribution of a random variable X is denoted by $p(X)$, and we write $p(X = x)$ or $p(x)$ the probability of x. We only work with discrete variables.

2 AN IDEALIZED EXPLANATION

Even though many variables $\mathbf{O}$ may be observed, the explanation can be focused only on a specific subset $\mathbf{E} \subseteq \mathbf{O}$. The state $\mathbf{E} = \mathbf{e}$ is then called *explanandum*.

The set of explanatory variables $\mathbf{H} \subsetneq \mathbf{V}$ can include both observed and unobserved variables, and an explanation is an assignment $\mathbf{H} = \mathbf{h}$ (compatible with $\mathbf{O} = \mathbf{o}$ for variables both in $\mathbf{H}$ and in $\mathbf{O}$).



We insist on the distinction between the explanandum $\mathbf{e}$ and the observations $\mathbf{o}$ (Chajewska and Halpern, 1997). Observations are all our knowledge about the current state of a system, and this might not coincide exactly with what we want explained. Consider for example the case where we wish to know why the grass is wet while we know it has been raining. We do not seek an explanation for why it has rained, only for why the grass is wet. A perfectly valid explanation is that the grass is wet because it is raining if no other factors can sufficiently explain the facts.

An algorithm respecting this should then determine, for each variable in $\mathbf{O}$, whether its observed state is relevant to explain $\mathbf{e}$, and for each unobserved variable in $\mathbf{V} \setminus \mathbf{O}$, whether knowing its state adds "explanatory power" to the proposed explanation. This excludes methods which marginalize out $\mathbf{O}$, preventing these variables from being part of an explanation.

To explain *why* a given system is observed in a given state, we must intuitively convey some information about the causal mechanisms that lead to the observation made. If we observe that it is raining and some explanation tells us that "it rains because the grass is wet," we do not find it a good explanation as it contradicts our understanding of how the system works; that the grass being wet cannot make it rain. Suppose we have an explanation $\mathbf{H} = \mathbf{h}$ for $\mathbf{E} = \mathbf{e}$: an intuitive interpretation of this result is that manually setting $\mathbf{H} = \mathbf{h}$ will be a favourable configuration to observe $\mathbf{E} = \mathbf{e}$. As Halpern and Pearl (2005) discuss, explanations need to be causal to be consistent with users' knowledge of the mechanisms of the system. It is therefore important that the explanations are given in a data-generating direction, such that users can infer interventional rules from the given explanations (for instance, "if I can make it rain somehow, then I know that the grass will be wet," as opposed to an impossible "let me make the grass wet so as to make it rain").

Causal methods are subject to the availability of causal information. In this paper, we extract the causal information from causal BNs, but in general, the approach is adaptable to any causal model that can predict the effect of interventions on certain variables.

In addition to assuming that the relationships between the variables $\mathbf{V}$ can be represented by a fully oriented causal BN, we assume that the corresponding joint probability distribution is faithful and causally sufficient (Pearl, 2000; Spirtes et al., 2001). *Faithfulness* of the distribution ensures that there is a unique graph whose arcs depict all (conditional) dependencies of the distribution, and only those. *Causal sufficiency* forbids hidden common causes for variables in $\mathbf{V}$, such that we can build a DAG whose arcs represent direct causation. Although most expert-designed BN are naturally oriented causally, the output of structure causal learning algorithms are often partially directed graphs and may need additional expert knowledge to be fully oriented.

To summarize, we wish our explanations to give us causal information by detailing the mechanisms that lead to the explanandum, using all the available information we have about the state of the network.

3 EXISTING METHODS

This section reviews and discusses some of the major techniques to find explanations.

3.1 MOST PROBABLE EXPLANATION & VARIANTS

A common noncausal measure of explanatory power is the conditional probability of the explanatory variables $\mathbf{H}$ given the explanandum $\mathbf{e}$. The *most probable explanation* (MPE) approach (Pearl, 1988) then considers $\mathbf{h}^* = \arg\max_{\mathbf{h}} p(\mathbf{h} \mid \mathbf{e})$ as the best explanation (or, alternatively, looks for the k best explanations by maximizing this probability). The explanandum $\mathbf{e}$ is in the case of MPE equal to the full set of observations $\mathbf{O} = \mathbf{o}$, and the set $\mathbf{H}$ is $\mathbf{V} \setminus \mathbf{E}$. This list can be long and uninformative because of lack of conciseness; moreover, it is hard to distinguish between long explanations, whose respective probabilities are low anyway and close to one another.

In the *partial abduction* approach (Shimony, 1991), the set of explanatory variables is a strict subset $\mathbf{H} \subsetneq \mathbf{V} \setminus \mathbf{E}$. The set of variables $\mathbf{X} = \mathbf{V} \setminus \mathbf{H} \setminus \mathbf{E}$ excluded from the explanation is then marginalized out before the maximum is computed: we look for $\arg\max_{\mathbf{h}} \sum_{\mathbf{x}} p(\mathbf{h}, \mathbf{x} \mid \mathbf{e})$. This is the *maximum a posteriori* (MAP) model approach. The excluded variables $\mathbf{X}$ are selected either by a user, or via automated analysis of the network. Automatically selecting the relevant explanatory variable is a nontrivial issue (Shimony, 1991).

Partial abduction is computationally more expensive than standard MPE, because it cannot be readily solved by message passing algorithms, but approximations exist (e.g., Park, 2002). On the other hand, it globally leads to more concise explanations than MPE.

Further efforts to make explanations more concise include de Campos et al. (2001), where the k most probable explanations are found and then simplified based on relevance and probabilistic criteria; and Henrion and Druzdzel (1990), where also partial assignments are allowed but only within a predefined tree that limits the set of possible explanations. An explanation is then a path from the root of the tree to a leaf, denoting variable assignments for each branch taken. This is known as *scenario-based explanation*. The best explanation is the one with the highest posterior probability.

There are several concerns with these approaches—MPE/MAP or scenario-based—maximizing some conditional probability of the explanatory variables (Chajewska and Halpern, 1997). First, they do not distinguish the explanandum and the observations, such that the additional state information that is not meant to be explained is excluded from a possible explanation. Furthermore, there is no distinction between observing an explanatory variable X in a certain state x, and forcing it to have the value x.[1] Thus, depending on the choice of the explanatory variables, the intuitive interpretation (as described in the previous section) stating that setting $\mathbf{H} = \mathbf{h}^*$ will be a favourable configuration for observing $\mathbf{E} = \mathbf{e}$ does not hold.

MPEs and, to a lesser extent, MAP model explanations, are not robust: little changes in the network will often change the result of the analysis, even though the changes occur in parts of the network largely independent of the explanandum (Chan and Darwiche, 2006). Common to the methods in this subsection is that they order explanations by $p(\mathbf{h}, \mathbf{e})$ (this is equivalent to $p(\mathbf{h} \,|\, \mathbf{e})$ as $p(\mathbf{e})$ is constant for a given $\mathbf{e}$): this joint probability cannot be considered alone to determine the explanatory power of $\mathbf{h}$ on $\mathbf{e}$. Some of these problems are illustrated by the experiments in section 5.

3.2 SE ANALYSIS

In SE analysis, Jensen (2001) additionally considers the sensitivity of an explanation $\mathbf{h}$ with respect to the explanandum. Less sensitive explanations ensure that little changes in the network's parameters will not lead to severely different explanations, so that the explanation is stable with respect to the specification of the network.

[1] The difference between observation and intervention is fundamental to causality and is best described with the example of Simpson's paradox in Pearl (2000), chap. 6.

SE analysis also works by comparing two explanations $\mathbf{h}_i$ and $\mathbf{h}_j$, usually with *Bayes' factor* or the *likelihood ratio* (Jeffreys, 1961): Bayes' factor $=$

$$\frac{\text{posterior ratio}}{\text{prior ratio}} = \frac{p(\mathbf{h}_i \,|\, \mathbf{e}) \,/\, p(\mathbf{h}_j \,|\, \mathbf{e})}{p(\mathbf{h}_i) \,/\, p(\mathbf{h}_j)} = \frac{p(\mathbf{e} \,|\, \mathbf{h}_i)}{p(\mathbf{e} \,|\, \mathbf{h}_j)}.$$

The empirical interpretation of Bayes' factor given by Jeffreys (1961) is that if it is less than 1 it is in favor of $\mathbf{h}_j$, if less than 3 it is a slight support for $\mathbf{h}_i$. If it is between 3 and 12, it is a positive support; and higher than 12, it is a strong support for $\mathbf{h}_i$.

In Yuan and Lu (2007), Bayes' factor is used to search for explanations consisting of only a few variables by ranking them by their Bayes' factor computed as the ratio between the probability of the explanation given the explanandum and its opposite:

$$\text{Bayes' factor} = \frac{p(\mathbf{h} \,|\, \mathbf{e})}{1 - p(\mathbf{h} \,|\, \mathbf{e})}.$$

An exhaustive search is performed over all subsets of the hypothesis, and the explanations are shown to be more concise in a sample network than MPE, Shimony's (1991) MAP, and the simplifications described by de Campos et al. (2001).

A similar criticism as before can be applied to these methods: additional observations are discarded, and the causal directionality is ignored in the selection of the relevant explanatory variables.

3.3 EXPLANATION TREES

The method of Flores (2005) constructs a set of best explanations while at the same time giving a preference for concise explanations, summarizing the results of the analysis in an *explanation tree*. We describe this method in more detail, as the causal information tree method (described in section 4) is based on a similar representation.

Definition 1 *An **explanation tree** for an explanandum $\mathbf{E} = \mathbf{e}$ is a tree in which every node X is an explanatory variable (with $X \in \mathbf{V} \setminus \mathbf{E}$), and each branch out of X is a specific instantiation $x \in \mathcal{X}$ of X. A path from the root to a leaf is then a series of assignments $X = x$, $Y = y$, $\cdots$, $Z = z$, summarized as $\mathbf{P} = \mathbf{p}$, which constitutes a full explanation.*

Flores's (2005) algorithm, summarized in Algorithm 1, builds such an explanation tree. Starting with an empty tree, the variable to use as the next node is selected to be the one that, given the explanandum, reduces the uncertainty the most in the rest of the explanatory variables according to some measure. The nodes that are on the path being grown are added to

a conditioning set, so that the part of the explanatory space they already account for is taken into account. Two stopping criteria are used to determine when to stop growing the tree: the minimum posterior probability β of the current branch, and the minimum amount of uncertainty reduction α that must be achieved by adding a new variable. Among all explanations represented by the final tree, the best one is the one with the largest posterior probability $p(\mathbf{p} \mid \mathbf{e})$.

Algorithm 1 Flores's (2005) Explanation Tree

1: **function** $T = \textsc{ExplanationTree}(\mathbf{H}, \mathbf{e}, \mathbf{p};\ \alpha, \beta)$
   Input: $\mathbf{H}$: set of explanatory variables
   $\mathbf{E} = \mathbf{e}$: explanandum
   $\mathbf{P} = \mathbf{p}$: path of variable assignments
   α, β : stopping criteria
   Output: T : an explanation tree
2: $X^* \leftarrow \arg\max_{X \in \mathbf{H}} \sum_{Y \in \mathbf{H}} \textsc{Inf}(X; Y \mid \mathbf{e}, \mathbf{p})$
3: **if** $\max_{Y \in \mathbf{H} \setminus X^*} \textsc{Inf}(X; Y \mid \mathbf{e}, \mathbf{p}) < \alpha$ **or** $p(\mathbf{p} \mid \mathbf{e}) < \beta$ **then**
4: **return** $\emptyset$
5: **end if**
6: $T \leftarrow$ new tree with root X^*
7: **for each** $x \in \text{domain}(X^*)$ **do**
8: $T' \leftarrow \textsc{ExplanationTree}(\mathbf{H} \setminus X^*, \mathbf{e}, \mathbf{p} \cup \{x\})$
9: add a branch x to T with subtree T' and
10: assign it the label $p(\mathbf{p}, x \mid \mathbf{e})$
11: **end for**
12: **return** T

The algorithm is parametrized with the measure of uncertainty reduction $\textsc{Inf}(X; Y \mid \mathbf{e}, \mathbf{p})$.[2] For our implementation, we used the *conditional mutual information*. The mutual information $I(X; Y \mid \mathbf{z})$ is a symmetrical measure of how much reduction in uncertainty about Y we get by knowing X in the context $\mathbf{Z} = \mathbf{z}$, and is defined as: $I(X; Y \mid \mathbf{z}) =$

$$\sum_{x \in \mathcal{X}} p(x \mid \mathbf{z}) \sum_{y \in \mathcal{Y}} p(y \mid x, \mathbf{z}) \log \frac{p(y \mid x, \mathbf{z})}{p(y \mid \mathbf{z})}. \qquad (1)$$

If X and Y are independent given $\mathbf{Z} = \mathbf{z}$, we have $I(X; Y \mid \mathbf{z}) = 0$. If X fully determines Y, then knowing one is enough to know the other and full information is shared.

Explanation trees are interesting in that they can present many mutually exclusive explanations in a compact form. Flores (2005) also argues that explanations as constructed by Algorithm 1 are reasonable and more sensible than (k-)MPE in the sense that on simple networks, the returned explanations are those that we expect. Four elements, however, are subject to discussion.

First, on line 2 of Algorithm 1, variables are added to the tree in order of how much information they provide about the remaining variables in the set of explanatory variables. But this does not measure the information of the added variables shared with the explanandum. Moreover, the explanandum actually grows as the tree is constructed, since there is no difference between the constructed path $\mathbf{p}$ and $\mathbf{e}$ at line 2. Thus, this maximization cannot be interpreted as selecting variables reducing the uncertainty in the explanandum.

Second, the algorithm makes no distinction between explanandum and observations. To try to fix this, we could either additionally condition on observations $\mathbf{O} = \mathbf{o}$, or marginalize out $\mathbf{O}$ altogether. The former case is no different from adding $\mathbf{o}$ to the explanandum $\mathbf{e}$, and the latter case excludes all $X \in \mathbf{O}$ from explanations, such that both cases are unsatisfactory.

Third, the criterion to choose the best explanation is the probability of the explanation path given the evidence $p(\mathbf{p} \mid \mathbf{e})$, and not how likely the system is to have produced the evidence we are trying to explain with a configuration $\mathbf{p}$, $p(\mathbf{e} \mid \mathbf{p})$. Both measures are linked, but since several explanations can cover almost an equal share of the explanation space and often only one will be included in the explanation tree, the criterion $p(\mathbf{p} \mid \mathbf{e})$ will miss explanations which could have explained the evidence well, but do not cover as large a fraction of the explanation space.

Fourth, causal considerations are ignored: there is no distinction between ancestors and descendants of a variables, such that we can get explanations of the type "it rains because the grass is wet."

From an end-user perspective though, trees are a good solution for representing several competing explanations compactly and readably. We introduce in section 4 a modified approach, which can address the issues discussed here.

4 CAUSAL EXPLANATION TREES

Like the previous method, causal explanation trees take advantage of a tree representation. The tree is grown so as to ensure that explanations in any path are causal: variables can be selected as explanatory only if they causally influence the explanandum.

Before defining the causal criterion used in this approach, we need to define the concept of *postintervention distribution* (Pearl, 2000, p. 72). A standard conditional probability of the form $p(\mathbf{e} \mid x)$ gives the probability (or probability density) of $\mathbf{e}$ when $X = x$ is observed. It does not represent, however, the probability of $\mathbf{e}$ if we manually force variable X to have value x. Causally, we are interested in the *intervention* on X, which we denote by $do(X = x)$, rather the observation of x. In causal BNs, the tool used to evaluate

[2] See Flores (2005) for additional cases where max at line 3 is replaced by min or avg, and Inf is the Gini index.

the effect of these conditionings is Pearl's (1995) *do*-calculus, which uses the structure of the causal graph to evaluate the postintervention distribution.

Definition 2 *Given a causal Bayesian network $\mathcal{B}$ in the sense of Pearl (2000, p. 23) over variables $\mathbf{V} = \{X_1, \cdots, X_d\}$, the **postintervention distribution** $p(\mathbf{v} \,|\, do(X_i = x_i'))$, also denoted $p(\mathbf{v} \,|\, \hat{x}_i')$, after an intervention $do(X_i = x_i')$ can be expressed as:*

$$p(x_1, \cdots, x_d \,|\, \hat{x}_i') = \begin{cases} \prod_{j \neq i} p(x_j \,|\, \mathbf{pa}_j) & \text{if } x_i = x_i', \\ 0 & \text{if } x_i \neq x_i', \end{cases} \quad (2)$$

where $\mathbf{pa}_j$ is the values of the graphical parents (i.e., direct causes) of the node X_j in $\mathcal{B}$.

The *truncated factorization* of (2) states that the probability distribution is computed as if the manipulated variable X_i had no incoming causal influence (i.e., no direct causes), and as if $p(X_i = x_i')$ had probability one. This makes sense, as forcing X_i to have a certain value effectively ignores its direct causes and "natural" distribution.

With this concept, we can now define the *causal information flow* (Ay and Polani, 2006), which will be our measure of causal contribution of explanatory variables towards our explanandum.

Definition 3 *The **causal information flow** from X to Y given the interventions $do(\mathbf{Z} = \mathbf{z})$, written $I(X \to Y \,|\, \hat{\mathbf{z}})$, is: $I(X \to Y \,|\, \hat{\mathbf{z}}) =$*

$$\sum_{x \in \mathcal{X}} p(x \,|\, \hat{\mathbf{z}}) \sum_{y \in \mathcal{Y}} p(y \,|\, \hat{x}, \hat{\mathbf{z}}) \log \frac{p(y \,|\, \hat{x}, \hat{\mathbf{z}})}{p^*(y \,|\, \hat{\mathbf{z}})}, \quad (3)$$

where $p^*(y \,|\, \hat{\mathbf{z}}) = \sum_{x' \in \mathcal{X}} p(x' \,|\, \hat{\mathbf{z}}) p(y \,|\, \hat{x}', \hat{\mathbf{z}})$.

The expression $I(X \to Y \,|\, \hat{\mathbf{z}})$ measures the amount of information flowing from X to Y if we intervene on $\mathbf{Z}$, setting it to $\mathbf{z}$ (i.e., if we block the causal flow on all paths going through $\mathbf{Z}$). Note that (3) is, in essence, similar to (1). For faithful probability distributions, $I(X \to Y \,|\, \hat{\mathbf{z}}) = 0$ if and only if all directed paths (if any) from X to Y go through $\mathbf{Z}$ in the corresponding causal graph. For binary variables, if Y is a deterministic function of X regardless of $\mathbf{Z}$, then $I(X \to Y \,|\, \hat{\mathbf{z}}) = 1$.

In our application, we use the causal information flow to decide which variable should be added to the tree being built. This is shown in Algorithm 2. At line 2, we use the *do*-conditioning on the already build path $\mathbf{p}$, and we allow inputting additional observed variables $\mathbf{O}$ in an additional conditioning set. We replace Y from (3) with the state of the explanandum e, suppress the corresponding summation and divide by the prior probability $p(e \,|\, \mathbf{o}, \hat{\mathbf{p}})$, so that we end up computing $I(X \to e \,|\, \mathbf{o}, \hat{\mathbf{p}}) =$

$$\sum_{x \in \mathcal{X}} \frac{p(x \,|\, \mathbf{o}, \hat{\mathbf{p}}) p(e \,|\, \mathbf{o}, \hat{x}, \hat{\mathbf{p}})}{p(e \,|\, \mathbf{o}, \hat{\mathbf{p}})} \log \frac{p(e \,|\, \mathbf{o}, \hat{x}, \hat{\mathbf{p}})}{\sum_{x' \in \mathcal{X}} p(x' \,|\, \mathbf{o}, \hat{\mathbf{p}}) p(e \,|\, \mathbf{o}, \hat{x}', \hat{\mathbf{p}})},$$

ensuring that the expected value $\mathbb{E}_E\big[I(X \to e \,|\, \mathbf{o}, \hat{\mathbf{p}})\big] = \sum_{e \in \mathcal{E}} p(e \,|\, \mathbf{o}, \hat{\mathbf{p}}) I(X \to e \,|\, \mathbf{o}, \hat{\mathbf{p}})$ equals $I(X \to E \,|\, \mathbf{o}, \hat{\mathbf{p}})$.

Using this criterion, the explanation tree is then built as follows: the root node is selected as being $\arg\max_X I(X \to e \,|\, \mathbf{o})$; i.e., the node which has the maximum information flow to the state of the explanandum. The important part is that we may condition on $\mathbf{o}$ without confusing observation and explanandum. Furthermore, we also allow selection of an observed variable $X \in \mathbf{O}$ in addition to unobserved variables, consistently with our desiderata. When $X \in \mathbf{O}$, it is observed and we know its value x. We must then compute the pointwise causal information flow from x to e, with $\mathbf{o}'$ being $\mathbf{o}$ without the observation $X = x$:

$$I(x \to e \,|\, \mathbf{o}', \hat{\mathbf{p}}) = \log \frac{p(e \,|\, \mathbf{o}', \hat{\mathbf{p}}, \hat{x})}{\sum_{x' \in \mathcal{X}} p(x' \,|\, \mathbf{o}', \hat{\mathbf{p}}) p(e \,|\, \mathbf{o}', \hat{x}', \hat{\mathbf{p}})}.$$

The tree is then grown recursively: for each possible value for X, a branch is added to the root. For each new leaf, the next explanatory variable is selected as being $\arg\max_Y I(Y \to e \,|\, \mathbf{o}, \hat{x})$, and so on, where the *do*-conditioning set always reflects the selected variable values from the root to the current leaf. We use only one stopping criterion, the minimum information flow α we accept as a causal information contribution. The algorithm furthermore allows explicitly to restrict the search set for explanatory variables $\mathbf{H}$ (defaulting to $\mathbf{V} \setminus \{E\}$). Finally, each leaf is labeled with $\log\big(p(e \,|\, \mathbf{o}, \hat{\mathbf{p}}) / p(e \,|\, \mathbf{o})\big)$ (where we make sure that variables selected in $\hat{\mathbf{p}}$ are removed from $\mathbf{o}$ if needed). This measures how much performing the interventions $\hat{\mathbf{p}}$ changes the probability of the explanandum (given the observations) with respect to the prior probability of the explanandum. Higher values indicate better explanations; negative values indicate that the probability of the explanandum actually decreases with the proposed explanation.

Using the information flow criterion brings us two advantages over standard (conditional) mutual information: first, we automatically only consider variables that can causally influence the explanandum. Second, when selecting the ith variable on a tree branch, we take into account the previously selected variables 1 through $i-1$ causally, as they enter the conditioning set of variables that have been intervened on.

In practice, computing a causal information flow of the type $I(X \to e \,|\, \mathbf{o}, \hat{\mathbf{p}})$ at line 2 of Algorithm 2 requires

Algorithm 2 Causal Explanation Tree

function $T = \textsc{CausalExplTree}(\mathbf{H}, \mathbf{o}, e, \hat{\mathbf{p}};\ \alpha)$
  Input: $\mathbf{H}$: set of explanatory variables
  $\mathbf{O} = \mathbf{o}$: observation set
  $E = e$: explanandum
  $\hat{\mathbf{p}}$: path of interventions
  α : stopping criterion
  Output: T : a causal explanation tree

$X^* \leftarrow \arg\max_{X \in \mathbf{H}} I(X \rightarrow e \,|\, \mathbf{o}, \hat{\mathbf{p}})$
if $I(X^* \rightarrow e \,|\, \mathbf{o}, \hat{\mathbf{p}}) < \alpha$ **then return** $\emptyset$
$T \leftarrow$ new tree with root X^*
for each $x \in \text{domain}(X^*)$ **do**
  $T' \leftarrow \textsc{CausalExplTree}(\mathbf{H} \setminus \{X^*\}, \mathbf{o}, e, \hat{\mathbf{p}} \cup \{\hat{x}\})$
  add a branch x to T with subtree T' and
  assign it the contribution $\log\big(p(e \,|\, \mathbf{o}, \hat{\mathbf{p}}, \hat{x}) / p(e \,|\, \mathbf{o})\big)$
end for
return T

to know the distributions $p(X \,|\, \mathbf{o}, \hat{\mathbf{p}})$, $p(E \,|\, \mathbf{o}, \hat{\mathbf{p}})$, and $p(E \,|\, \mathbf{o}, \hat{X}, \hat{\mathbf{p}})$ (i.e., $p(E \,|\, \mathbf{o}, \hat{x}, \hat{\mathbf{p}})$ for all $x \in \mathcal{X}$). Additionally, $p(e \,|\, \mathbf{o})$ is needed to label the leaves, but as it is only dependent on e and $\mathbf{o}$, we compute it only once. We can further avoid unnecessary computations by using the graphical reachability criterion from a candidate node X to E, blocking paths going through $\mathbf{O} \cup \mathbf{P}$. The inference steps were implemented using the factor graph message-passing algorithm (Frey et al., 2001).

The complexity of this algorithm, in terms of number of calls to an inference engine per node in the constructed tree, is $\mathcal{O}(nd)$, where n is the number of explanatory variables $|\mathbf{H}|$ and d is the average domain size of the variables, e.g., 2 for binary variables. For comparison, Flores's (2005) approach is $\mathcal{O}(n^2 d^2)$.

5 EXPERIMENTS

We compare causal explanation trees (CET) with parameter $\alpha = 0$ to Most Probable Explanation (MPE), Bayes' factor (BF) following Yuan and Lu (2007), and standard (noncausal) explanation trees (ET) with parameters $\alpha = 0.02$ and $\beta = 0$. We test the approach on three simple networks[3] to compare the relevance of explanations. A more extended version of these experiments and comments can be found in Nielsen (2007).

Drug (Figure 1). This network comes from Pearl (2000, chap. 6). It represents the outcome of an experiment designed to check the efficiency of a new drug on male and female patients. The males have a natural recovery rate of 70%; taking the drug decreases it to 60%. Similarly, 30% of females recover naturally, but only 20% when given the drug. Thus, both the absence of drug and being a male can explain a good recovery rate.

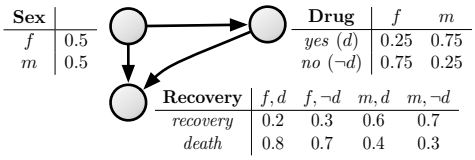
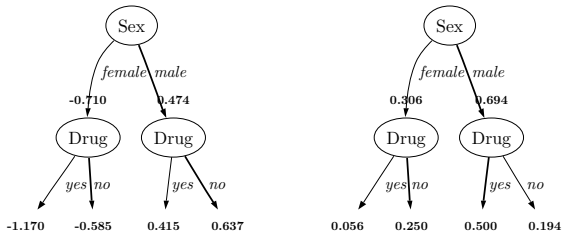


Figure 1: The DRUG network.



(*a*) Causal explanation tree (*b*) Explanation tree

(*c*) MPE: $p(Sex = m,\ Drug = yes \,|\, Recovery = rec.) = 0.5$
(*d*) BF: $\text{BF}(Sex = m) = 2.27$
$\text{BF}(Sex = m,\ Drug = no) = 1.68$
$\text{BF}(Drug = yes) = 1.25$

Figure 2: DRUG: explain $Recovery = recovery$.

In Figure 2, we try to explain a recovery. All approaches correctly realize that $Sex = m$ largely accounts for the recovery. However, ET selects $Sex = m \wedge Drug = yes$ as the best explanation according to the leaves' labels, just like MPE. This contradicts the natural idea of explanation, since the drug has a negative impact on the recovery. CET labels the leaves more sensibly: branches where the drug was not given have a higher rank. Moreover, the branches where $Sex = f$ have a negative label, indicating that they actually decrease the probability of recovery. Although the first two BF explanations are sensible, the third one is mistakenly selects $Drug = yes$ as an explanation.

Academe (Figure 3). This network depicts the relationships between various marks given to students following a course. The *Final mark* is determined by some *Other* outside factors and an intermediate mark (*T.P. mark*), which is in turn determined by the student's abilities in *Theory* and *Practice* as well as *Extra* curricular activities in this tested subject.

In Figure 4, the explanandum was set to *Final mark* = *fail*; i.e., we want to explain why a student failed the course. *T.P. mark* and *Global mark* have been excluded from the possible explanatory variables in the two tree algorithms as they are modeling artifacts.

ET tells us that *Theory* = *bad* is the best explanation. We could have expected *Practice* to also be part of

[3]The conditional probability tables have been omitted in the figures of the two larger BNs due to lack of space and can be found at `http://www.zurich.ibm.com/~uln/causalexpl/`.

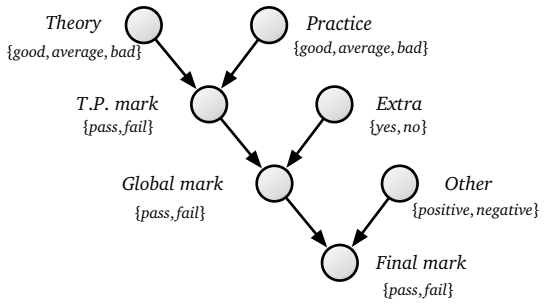
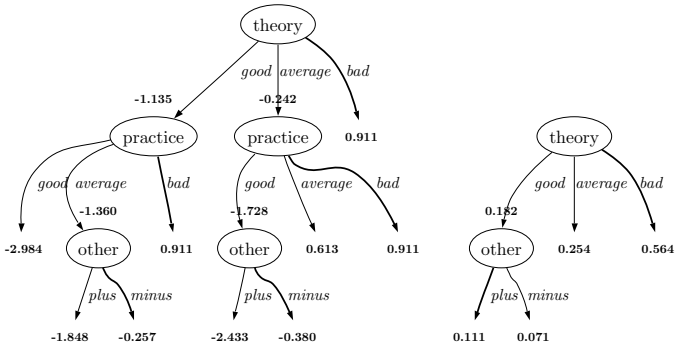


Figure 3: The ACADEME network.



(*a*) Causal explanation tree (*b*) Explanation tree

(*c*) MPE: $p(\mathit{Theory} = \mathit{bad},\ \mathit{T.P.\ mark} = \mathit{fail},\ \mathit{Global\ mark} = \mathit{fail},\ \mathit{Extra} = \mathit{no},\ \mathit{Other} = \mathit{positive},\ \mathit{Practice} = \mathit{good} \mid \mathit{Final\ mark} = \mathit{fail}) = 0.208$

(*d*) BF: $\mathrm{BF}(\mathit{Theory} = \mathit{bad}) = 3.02$
$\mathrm{BF}(\mathit{Theory} = \mathit{bad},\ \mathit{Extra} = \mathit{no}) = 2.78$
$\mathrm{BF}(\mathit{Theory} = \mathit{bad},\ \mathit{Other} = \mathit{negative}) = 2.53$

Figure 4: ACADEME: explain $\mathit{Final\ mark} = \mathit{fail}$.

alternate explanations, as it influences the final mark very similarly to *Theory*. This is what CET does, including *Practice* to explain the final failure when *Theory* is *average* or *good*. MPE includes $\mathit{Practice} = \mathit{good}$ in its long list of states, which does not seem intuitively likely. BF provides more concise explanations, but, like ET, ignores *Practice* altogether, although a bad practice can account for failure equally well.

Asia (Figure 5). This network (Lauritzen and Spiegelhalter, 1988) models the relationships between two indicators, X-ray results and dyspnea, of severe diseases for a person. Tuberculosis (more likely if a visit to Asia occurred) and lung cancer (more likely when the person smokes) both increase abnormal X-ray results and dyspnea; bronchitis also causes increased dyspnea. *TbOrCa* is a modeling artifact, excluded from the the analysis in the two tree algorithms.

In Figure 6, we try to explain abnormal X-ray results. Whereas both tree algorithms select a *Lung cancer* as the best explanation, they differ on how to explain

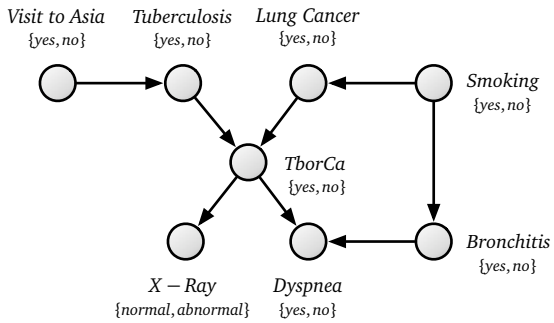
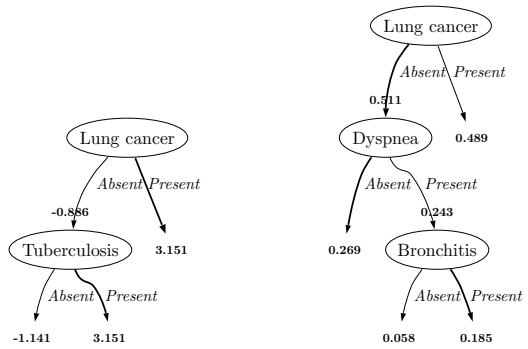


Figure 5: The ASIA network.

when it is *absent*: CET selects, justifiably, *Tuberculosis*, but ET uses *Dyspnea* and then *Bronchitis*, which are not causes of *X-ray* and cannot explain it, especially not when we know that no lung cancer is present. MPE surprisingly excludes a visit to Asia, when this is expected to make abnormal X-rays more likely through *Tuberculosis*. BF, while still providing very concise explanations, is stuck on the middle node *TbOrCa* and, like ET, ignore the important *Tuberculosis*.



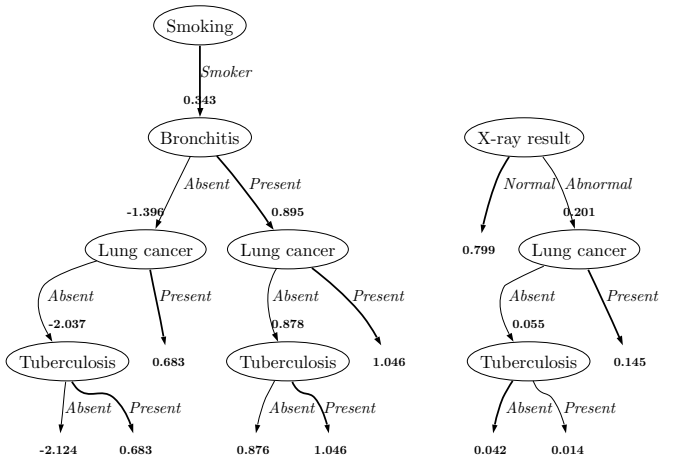
(*a*) Causal explanation tree (*b*) Explanation tree

(*c*) MPE: $p(\mathit{Bronchitis} = \mathit{yes},\ \mathit{Dyspnea} = \mathit{yes},\ \mathit{Lung\ cancer} = \mathit{yes},\ \mathit{TbOrCa} = \mathit{yes},\ \mathit{Smoker} = \mathit{yes},\ \mathit{Tuberculosis} = \mathit{yes},\ \mathit{Visit\ to\ Asia} = \mathit{no} \mid \mathit{X\text{-}ray} = \mathit{abnormal}) = 0.24$

(*d*) BF: $\mathrm{BF}(\mathit{TbOrCa} = \mathit{yes}) = 19.60$
$\mathrm{BF}(\mathit{TbOrCa} = \mathit{yes},\ \mathit{Visit\ to\ Asia} = \mathit{no}) = 19.21$
$\mathrm{BF}(\mathit{TbOrCa} = \mathit{yes},\ \mathit{Lung\ cancer} = \mathit{yes}) = 16.42$

Figure 6: ASIA: explain $\mathit{X\text{-}ray} = \mathit{abnormal}$.

In Figure 7, we try to explain the presence of dyspnea for a smoker. While CET is still able to select *Smoker* as explanatory variable, ET can only add this observation to the explanandum and thus cannot select it. Instead, the best explanation according to ET is normal X-rays, which does not seem very likely. Although ET does select the important *Lung cancer* and *Tuberculosis*, it ignores the largest factor according to BF and CET, namely *Bronchitis*. Here too, CET selects the more intuitively interpretable explanations.



(*a*) Causal explanation tree (*b*) Explanation tree

(*c*) MPE: $p(Bronchitis = yes,\ X\text{-}ray = normal,\ Lung\ cancer = no,\ TbOrCa = no,\ Smoker = yes,\ Tuberculosis = no,\ Visit\ to\ Asia = no \mid Dyspnea = yes) = 0.46$

(*d*) BF: $\mathrm{BF}(Bronchitis = yes) = 6.14$
$\mathrm{BF}(Bronchitis = yes,\ Visit\ to\ Asia = no) = 5.89$
$\mathrm{BF}(Bronchitis = yes,\ Tuberculosis = no) = 5.84$

Figure 7: ASIA: explain $Dyspnea = yes \mid Smoker = yes$.

6 CONCLUSION

We have presented an approach to explanation in causal BNs, causal explanation trees. Explanations are presented as a tree, compactly representing several explanations and making it more readable than a (possibly long) list. Assuming that the BN is causal allows us to use the causal information flow criterion to build the tree. This leads to more sensible explanations, in that we only explain a given state with variables that can causally influence it. The approach makes an explicit distinction between observation and explanandum. This lets the user input all available knowledge about the network as observation, while still focusing on explaining one of them and allowing the observed variables to be selected as part of a good explanation. The algorithm labels the leaves so as to reflect how a proposed explanation changes the probability of the explanandum, making the tree easy to interpret.

Causal explanation trees, unlike other techniques, do not condition on the explanandum to maximize the probability of the explanatory variables $p(\mathbf{h} \mid \mathbf{e})$, but focus on $p(\mathbf{e} \mid \mathbf{h})$ instead by means of the causal information flow. This allows it to compare favorably to MPE, Bayes' factor, and Flores's (2005) noncausal explanation trees on the tested networks because the returned explanations are intuitive, and those with a positive label in the tree ensure that they increase the probability of the explanandum.

References

Ay, N. & Polani, D. (2006). Information flows in causal networks. Technical report, Max Planck Institute for Mathematics in the Sciences.

Chajewska, U. & Halpern, J. Y. (1997). Defining explanation in probabilistic systems. In: *UAI-97*, pages 62–71. Morgan Kaufmann.

Chan, H. & Darwiche, A. (2006). On the robustness of most probable explanations. In: *UAI-06*.

de Campos, L. M., Gémez, J. A., & Moral, S. (2001). Simplifying explanations in Bayesian belief networks. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 9*.

Flores, M. J. (2005). *Bayesian networks Inference: Advanced algorithms for triangulation and partial abduction*. PhD thesis, Universidad De Castilla-La Mancha.

Frey, B. J., Kschischang, F. R., & Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47.

Halpern, J. Y. & Pearl, J. (2005). Causes and explanations: A structural-model approach. Part II: Explanations. *The British Journal for the Philosophy of Science*.

Henrion, M. & Druzdzel, M. J. (1990). Qualitative propagation and scenario-based scheme for exploiting probabilistic reasoning. In: *UAI*, pages 17–32.

Jeffreys, H. (1961). *Theory of Probability*. Oxford University Press.

Jensen, F. V. (2001). *Bayesian Networks and Decision Graphs*. Springer.

Lacave, C. & Diez, F. J. (2002). A review of explanation methods of Bayesian networks. *Knowledge Engineering Review*, 17(2):107–127.

Lauritzen, S. L. & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50:157–224.

Nielsen, U. H. (2007). On causal explanations in Bayesian networks. Master's thesis, IT University of Copenhagen.

Park, J. D. (2002). Map complexity results and approximation methods. In: *UAI-02*, pages 388–396.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Pearl, J. (1995). Causal diagrams for empirical research. *Biometrika*, 82(4):669–709.

Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

Shimony, S. E. (1991). Explanation, irrelevance, and statistical independence. In: *AAAI*, pages 482–487.

Spirtes, P., Glymour, C., & Scheines, R. (2001). *Causation, Prediction, and Search, Second Edition*. The MIT Press.

Yuan, C. & Lu, T.-C. (2007). Finding explanations in bayesian networks. In: *The 18th International Workshop on Principles of Diagnosis*.

On the Conditional Independence Implication Problem: A Lattice-Theoretic Approach

Mathias Niepert
Department of Computer Science
Indiana University
Bloomington, IN, USA
mniepert@cs.indiana.edu

Dirk Van Gucht
Department of Computer Science
Indiana University
Bloomington, IN, USA
vgucht@cs.indiana.edu

Marc Gyssens
Department WNI
Hasselt University & Transnational
University of Limburg, Belgium
marc.gyssens@uhasselt.be

Abstract

A lattice-theoretic framework is introduced that permits the study of the conditional independence (CI) implication problem relative to the class of discrete probability measures. Semi-lattices are associated with CI statements and a finite, sound and complete inference system relative to semi-lattice inclusions is presented. This system is shown to be (1) sound and complete for saturated CI statements, (2) complete for general CI statements, and (3) sound and complete for stable CI statements. These results yield a criterion that can be used to falsify instances of the implication problem and several heuristics are derived that approximate this "lattice-exclusion" criterion in polynomial time. Finally, we provide experimental results that relate our work to results obtained from other existing inference algorithms.

1 Introduction

Conditional independence is an important concept in many calculi for dealing with knowledge and uncertainty in artificial intelligence. The notion plays a fundamental role for learning and reasoning in probabilistic systems which are successfully employed in areas such as computer vision, computational biology, and robotics. Hence, new theoretical findings and algorithmic improvements have the potential to impact many fields of research. A central issue for reasoning about conditional independence is the *probabilistic conditional independence implication problem*, that is, to decide whether a CI statement is entailed by a set of other CI statements relative to the class of discrete probability measures. While it remains open whether this problem is decidable, it is known that there exists no finite, sound and complete inference system (Studený [8]). However, there exist finite sound inference systems that have attracted special interest. The most prominent is the *semi-graphoid* axiom system which was introduced as a set of sound inference rules relative to the class of discrete probability measures (Pearl [6]). One of the main contributions of this paper is to extend the semi-graphoids to a *finite* inference system, denoted by $\mathcal{A}$, which we will show to be (1) sound and complete for saturated CI statements, (2) complete for general CI statements, and (3) sound and complete for stable CI statements (de Waal and van der Gaag [2]), all relative to the class of discrete probability measures.

The techniques we use to obtain these results are made possible through the introduction of a *lattice-theoretic framework*. In this approach, semi-lattices are associated with conditional independence statements, and $\mathcal{A}$ is shown to be sound and complete relative to certain inclusion relationships on these semi-lattices. To make the connection between this framework and the conditional independence implication problem, we first link the latter to an addition-based version of the problem. In particular, we introduce the *additive implication problem* for CI statements relative to certain classes of real-valued functions and specify properties of these classes that guarantee soundness and completeness, respectively, of $\mathcal{A}$ for the implication problem. Through the concept of *multi-information functions* induced by probability measures (Studený [9]), we link the additive implication problem for this class of functions to the probabilistic CI implication problem.

The combination of the lattice-inclusion techniques and the completeness result for conditional independence statements allows us to derive criteria that can be used to falsify instances of the implication problem. We show experimentally that these criteria, some of which can be tested for in polynomial time, work very effectively, and we relate the experimental results to those obtained from a *racing algorithm* introduced by Bouckaert and Studený [1].

2 CI Statements and System $\mathcal{A}$

We define CI statements and introduce the finite inference system $\mathcal{A}$ for reasoning about the conditional independence implication problem. We will often write AB for the union $A \cup B$, ab for the set $\{a, b\}$, and a for the singleton set $\{a\}$ whenever the interpretation is clear from the context. Throughout the paper, S denotes a finite implicit set of statistical variables.

Definition 2.1. The expression $I(A, B|C)$ where A, B, and C are pairwise disjoint subsets of S is called a *conditional independence (CI) statement*. If $ABC = S$ we say that $I(A, B|C)$ is *saturated*. If either $A = \emptyset$ and/or $B = \emptyset$ we say that $I(A, B|C)$ is *trivial*.

$I(A, \emptyset\|C)$	**Triviality**
$I(A, B\|C) \rightarrow I(B, A\|C)$	**Symmetry**
$I(A, BD\|C) \rightarrow I(A, D\|C)$	**Decomposition**
$I(A, B\|CD) \wedge I(A, D\|C) \rightarrow I(A, BD\|C)$	**Contraction**
$I(A, B\|C) \rightarrow I(A, B\|CD)$	**Strong union**
$I(A, B\|C) \wedge I(D, E\|AC) \wedge I(D, E\|BC) \rightarrow I(D, E\|C)$	**Strong contraction**

Figure 1: The inference rules of system $\mathcal{A}$.

The set of inference rules in Figure 1 will be denoted by $\mathcal{A}$. The *triviality*, *symmetry*, *decomposition*, and *contraction* rules are part of the semi-graphoid axioms (Geiger [6]). *Strong union* and *strong contraction* are two additional inference rules. Note that *strong union* is not a sound inference rule relative to the class of discrete probability measures. The derivability of a CI statement c from a set of CI statements $\mathcal{C}$ under the inference rules of system $\mathcal{A}$ is denoted by $\mathcal{C} \vdash c$. The *closure* of $\mathcal{C}$ under $\mathcal{A}$, denoted $\mathcal{C}^+$, is the set $\{c \mid \mathcal{C} \vdash c\}$.

Lemma 2.2 (de Waal and van der Gaag [2])**.** *The inference rule* composition

$I(A, B|C) \wedge I(A, D|C) \rightarrow I(A, BD|C)$ **Composition**

can be derived using strong union *and* contraction.

3 Lattice-Theoretic Framework

First, we introduce the lattice-theoretic framework which is at the core of the theory developed in this paper. The approach we take is made possible through the association of conditional independence statements with semi-lattices. In this section, we prove that inference system $\mathcal{A}$ is sound and complete relative to specific semi-lattice inclusions. This result forms the backbone of our work on the conditional independence implication problem.

3.1 Semi-Lattices of CI Statements

Given two subsets A and B of S, we will write $[A, B]$ for the lattice $\{U \mid A \subseteq U \ \& \ U \subseteq B\}$. We will now associate semi-lattices with conditional independence statements.

Definition 3.1. Let $I(A, B|C)$ be a CI statement. The *semi-lattice* of $I(A, B|C)$ is defined by $\mathcal{L}(A, B|C) = [C, S] - ([A, S] \cup [B, S])$.

We will often write $\mathcal{L}(c)$ to denote the semi-lattice of a conditional independence statement c, and $\mathcal{L}(\mathcal{C})$ to denote the union of semi-lattices, $\bigcup_{c' \in \mathcal{C}} \mathcal{L}(c')$, of a set of conditional independence statements $\mathcal{C}$. Using the notion of *witnesses* of a conditional independence statement, we can rewrite the associated semi-lattice as a difference-free union of lattices.

Definition 3.2. Let $I(A, B|C)$ be a CI statement. The set of all witness sets of $I(A, B|C)$ is defined as $\mathcal{W}(A, B|C) = \{\{a, b\} \mid a \in A \text{ and } b \in B\}$.

Note that if $I(A, B|C)$ is trivial, then $\mathcal{W}(A, B|C) = \emptyset$.

Lemma 3.3. *Let $c = I(A, B|C)$ be a CI statement. Then $\mathcal{L}(c) = \bigcup_{W \in \mathcal{W}(c)} [C, \overline{W}]$.*

Example 3.4. Let $S = \{a, b, c, d\}$ and let $I(bc, d|a)$ be a CI statement. Then, $\mathcal{L}(bc, d|a) = [a, S] - ([bc, S] \cup [d, S]) = \{a, ab, ac\}$. Furthermore, $\mathcal{W}(bc, d|a) = \{bd, cd\}$ and, therefore, $\mathcal{L}(bc, d|a) = [a, ac] \cup [a, ab] = \{a, ab, ac\}$, using Lemma 3.3.

3.2 Soundness and Completeness of Inference System $\mathcal{A}$ for Semi-Lattice Inclusion

We will prove that system $\mathcal{A}$ is sound and complete relative to semi-lattice inclusion. First, we show that if a CI statement can be derived from a set of CI statements under $\mathcal{A}$, then we have a set inclusion relationship between their associated semi-lattices.

Proposition 3.5. *Let $\mathcal{C}$ be a set of CI statements, and let c be a CI statement. If $\mathcal{C} \vdash c$, then $\mathcal{L}(\mathcal{C}) \supseteq \mathcal{L}(c)$.*

Proof. We prove the statement for strong contraction. The proofs for the other inference rules in $\mathcal{A}$ are analogous and are omitted. Let $U \in \mathcal{L}(D, E|C)$. Then $U \supseteq C$. If $U \supseteq A$, then $U \in \mathcal{L}(D, E|AC)$. If $U \supseteq B$, then $U \in \mathcal{L}(E, D|BC)$. If $U \not\supseteq A$ and $U \not\supseteq B$, then $U \in \mathcal{L}(A, B|C)$. □

A CI statement can be equivalent to a set of other CI statements with respect to the inference system $\mathcal{A}$. The following definition of a *witness decomposition* of a CI statement is aimed to prove this property.

Definition 3.6. The *witness decomposition* of the CI statement $I(A, B|C)$ is defined by $wdec(A, B|C) := \{I(a, b|C) \mid a \in A \text{ and } b \in B\}$.

A useful property of the witness decomposition of a CI statement is that its closure under $\mathcal{A}$ is the same as the closure of the CI statement itself. In addition, the semi-lattice of a CI statement is equal to the semi-lattice of its witness decomposition.

Proposition 3.7. *Let c be a CI statement. (1) $\{c\}^+ = wdec(c)^+$; and (2) $\mathcal{L}(c) = \bigcup_{c' \in wdec(c)} \mathcal{L}(c')$.*

Proof. To prove the first statement, let $c = I(A, B|C)$ and $I(a, b|C) \in wdec(c)$. Then $I(a, b|C)$ can be derived from $I(A, B|C)$ by applications of the *decomposition* rule. Hence, $wdec(c)^+ \subseteq \{c\}^+$. By Definition 3.6 we know that for every $a \in A$ and for all $b \in B$ one has $I(a, b|C) \in wdec(c)$. By repeatedly applying *composition*, we can infer the CI statement $I(a, B|C)$. Hence, for all $a \in A$, one has $I(a, B|C) \in wdec(c)^+$ and by *symmetry* $I(B, a|C) \in wdec(c)^+$. Again, by applying *composition* repeatedly, we can infer $I(B, A|C)$ and by *symmetry* $I(A, B|C)$. Hence, $\{c\}^+ \subseteq wdec(c)^+$.

To prove the second statement, let $I(a, b|C) \in wdec(c)$ and $W = \{a, b\}$. Then $\mathcal{L}(a, b|C) = [C, \overline{W}]$. The statement now follows directly from Definition 3.6 and Lemma 3.3. □

We are now in the position to prove the main result concerning the soundness and completeness of the inference system $\mathcal{A}$ for semi-lattice inclusion.

Theorem 3.8. *Let $\mathcal{C}$ be a set of CI statements, and let c be a CI statement. Then $\mathcal{C} \vdash c$ if and only if $\mathcal{L}(\mathcal{C}) \supseteq \mathcal{L}(c)$.*

Proof. We already know by Proposition 3.5 that if $\mathcal{C} \vdash c$ then $\mathcal{L}(\mathcal{C}) \supseteq \mathcal{L}(c)$. We now proceed to show the other direction. Let us denote $wdec(\mathcal{C}) = \bigcup_{c' \in \mathcal{C}} wdec(c')$ and let $I(a, b|C) \in wdec(c)$ with $W = \{a, b\}$. From the assumption $\mathcal{L}(\mathcal{C}) \supseteq \mathcal{L}(c)$ and Proposition 3.7(2) it follows that $\mathcal{L}(\mathcal{C}) \supseteq \mathcal{L}(a, b|C)$ (1). By Proposition 3.7(1) it suffices to show that $I(a, b|C) \in wdec(\mathcal{C})^+$. However, we will prove the stronger statement $\forall V \in [C, \overline{W}] : I(a, b|V) \in wdec(\mathcal{C})^+$ by downward induction on the lattice $[C, \overline{W}]$.

For the base case we need to show that $I(a, b|\overline{W}) \in wdec(\mathcal{C})^+$. By (1) $\overline{W}$ is in $\mathcal{L}(\mathcal{C})$. Hence, by Proposition 3.7(1), there exists a CI statement $I(a, b|C') \in wdec(\mathcal{C})$ such that $\overline{W} \in \mathcal{L}(a, b|C')$. Now, since $C' \subseteq \overline{W}$, we can derive $I(a, b|\overline{W})$ through *strong union*.

For the induction step, let $C \subseteq V \subset \overline{W}$. The induction hypothesis states that for all V' with $V \subset V' \subseteq \overline{W}$ one has $I(a, b|V') \in wdec(\mathcal{C})^+$. By (1) V is in $\mathcal{L}(\mathcal{C})$. Hence, by Proposition 3.7(1), there exists a CI statement $I(a', b'|C') \in wdec(\mathcal{C})$ such that $V \in \mathcal{L}(a', b'|C')$. Since $C' \subseteq V$ we can use *strong union* to derive $I(a', b'|V)$. Let $W' = \{a', b'\}$. Note that $W' \cap V = \emptyset$. We distinguish three cases:

- $W' = W$. Then we are done.
- Exactly one of the two elements in W' is not in W. Without loss of generality let this element be b'. Then we can use *contraction* on the statements $I(a, b'|V)$ and $I(a, b|Vb')$ (the latter is in $wdec(\mathcal{C})^+$ by the induction hypothesis) to derive $I(a, b'b|V)$, and finally *decomposition* to derive $I(a, b|V)$.
- Both elements in W' are not in W. We can use *strong contraction* on the statements $I(a', b'|V)$, $I(a, b|Va')$, and $I(a, b|Vb')$ (the latter two are in $wdec(\mathcal{C})^+$ by the induction hypothesis) to derive $I(a, b|V)$.

This concludes the proof. □

Example 3.9. Let $S = \{a, b, c, d\}$, let $\mathcal{C} = \{I(a, b|\emptyset), I(c, d|a), I(c, d|b)\}$ and let $c = I(c, d|\emptyset)$. We can derive c from $\mathcal{C}$ using the inference rule *strong contraction*. In addition, $\mathcal{L}(\mathcal{C}) = \{\emptyset, c, d, cd\} \cup \{a, ab\} \cup \{b, ab\} = \{\emptyset, a, b, c, d, ab, cd\}$ and $\mathcal{L}(c) = \{\emptyset, a, b, ab\}$, and, therefore, $\mathcal{L}(\mathcal{C}) \supseteq \mathcal{L}(c)$.

4 The Additive Implication Problem for CI Statements

An important result in the study of the implication problem relative to the class of discrete probability measures was gained by Studený who linked it to an additive implication problem (Studený [9]). More specifically, it was shown that for every CI statement $I(A, B|C)$, a discrete probability measure P satisfies $I(A, B|C)$ if and only if the *multi-information function*[1] M_P induced by P satisfies the equality $M_P(C) + M_P(ABC) = M_P(AC) + M_P(BC)$. Thus, the *multiplication-based* probabilistic CI implication problem was related to an *addition-based* implication problem. It is this duality that is at the basis of the results developed in this section. However, rather than immediately focusing on specific classes of *multi-information functions*, which is what we pursue in Section 6, we first consider the additive implication problem for CI statements relative to arbitrary classes of real-valued functions.

By a *real-valued function*, we will always mean a function $F : 2^S \to \mathbf{R}$, i.e., a function that maps each subset of S into a real number.

Definition 4.1. Let $I(A, B|C)$ be a CI statement, and let F be a real-valued function. We say that F *a-satisfies* $I(A, B|C)$, and write $\models^a_F I(A, B|C)$, if $F(C) + F(ABC) = F(AC) + F(BC)$.

[1]The multi-information function of a probability measure will be formally defined in Section 6.

Relative to the notion of *a-satisfaction*, we can now define the *additive implication problem* for conditional independence statements.

Definition 4.2 (Additive implication problem)**.** Let $\mathcal{C}$ be a set of CI statements, let c be a CI statement, and let $\mathcal{F}$ be a class of real-valued functions. We say that $\mathcal{C}$ *a-implies* c relative to $\mathcal{F}$, and write $\mathcal{C} \models^a_{\mathcal{F}} c$, if each function $F \in \mathcal{F}$ that *a-satisfies* the CI statements in $\mathcal{C}$ also *a-satisfies* the CI statement c.

We now define the notion of density of a real-valued function. The density is again a real-valued function and plays a crucial role in reasoning about additive implication problems.

Definition 4.3. Let F be a real-valued function. The *density*[2] of F is the real-valued function ΔF defined by $\Delta F(X) = \sum_{X \subseteq U \subseteq S} (-1)^{|U|-|X|} F(U)$, for each $X \subseteq S$.

The following relationship between a real-valued function and its *density* justifies the name.

Proposition 4.4. *Let F be a real-valued function. Then, for each $X \subseteq S$, $F(X) = \sum_{X \subseteq U \subseteq S} \Delta F(U)$.*

The *a-satisfaction* of a real-valued function for a CI statement can be characterized in terms of an equation involving its density function. This characterization is central in developing our results and is a special case of a more general result by Sayrafi and Van Gucht who used it in their study of the *frequent itemset mining problem* (Sayrafi and Van Gucht [7]).

Proposition 4.5. *Let $I(A, B|C)$ be a CI statement and and let F be a real-valued function. Then, $\models^a_F I(A, B|C)$ if and only if $\sum_{U \in \mathcal{L}(A,B|C)} \Delta F(U) = 0$.*

5 Properties of Classes of Functions - Soundness and Completeness

In this section we study properties of classes of real-valued functions that guarantee soundness and completeness of $\mathcal{A}$, respectively, for the additive implication problem. How these results relate to probabilistic conditional independence implication will become clear in Section 7 and Section 8.

5.1 Soundness

First, we define the notion of soundness of system $\mathcal{A}$ for a given class of real-valued functions.

Definition 5.1 (Soundness)**.** Let $\mathcal{F}$ be a class of real-valued functions. We say that $\mathcal{A}$ is *sound* relative to $\mathcal{F}$ if, for each set $\mathcal{C}$ of CI statements and each CI statement c, we have that $\mathcal{C} \vdash c$ implies $\mathcal{C} \models^a_{\mathcal{F}} c$.

[2]What we call the *density* is sometimes referred to as the *Möbius inversion* of a real-valued function.

In order to characterize soundness we introduce the following property of classes of real-valued functions.

Definition 5.2 (Zero-density property)**.** Let $\mathcal{F}$ be a class of real-valued functions. We say that $\mathcal{F}$ has the *zero-density property* if, for each $F \in \mathcal{F}$, for each CI statement c, and for each $U \in \mathcal{L}(c)$, one has that if $\models^a_F c$, then $\Delta F(U) = 0$.

We can now provide various characterizations of the soundness of inference system $\mathcal{A}$ for the additive implication problem for CI statements.

Theorem 5.3. *Let $\mathcal{F}$ be a class of real-valued functions. Then, the following statements are equivalent:*

(1) Strong union and decomposition are sound inference rules relative to $\mathcal{F}$ for the additive implication problem;

(2) $\mathcal{F}$ has the zero-density property; and

(3) $\mathcal{A}$ is sound relative to $\mathcal{F}$ for the additive implication problem.

Proof. We first prove that statement (1) implies statement (2). Let $F \in \mathcal{F}$, let $I(A, B|C)$ be a CI statement, and assume $\models^a_F I(A, B|C)$. We now show that $\Delta F(V) = 0$ for each $V \in \mathcal{L}(A, B|C)$. The proof goes by downward induction on the semi-lattice $\mathcal{L}(A, B|C)$. First, we observe that by Lemma 3.3, $\mathcal{L}(A, B|C) = \bigcup_{W \in \mathcal{W}(A,B|C)} [C, \overline{W}]$. Hence, for the base case we must prove that $\Delta F(\overline{W}) = 0$ for each $W \in \mathcal{W}(A, B|C)$. Let $W = \{a, b\} \in \mathcal{W}(A, B|C)$. $I(a, b|C)$ is derivable from $I(A, B|C)$ using the inference rule *decomposition* and therefore $\models^a_F I(a, b|C)$. Since *strong union* is assumed sound and $C \subseteq \overline{W}$ it follows that $\models^a_F I(a, b|\overline{W})$. Since $\mathcal{L}(a, b|\overline{W}) = \{\overline{W}\}$ we can invoke Proposition 4.5 to conclude that $\Delta F(\overline{W}) = 0$. For the induction step, let $V \in \mathcal{L}(A, B|C)$. The induction hypothesis states that $\Delta F(U) = 0$ for all $U \in \mathcal{L}(A, B|C)$ that are strict supersets of V. Similar to the base case, we can infer that $\models^a_F I(A', B'|V)$ with A', B', and V pairwise disjoint, $A' \subseteq A$, $B' \subseteq B$, and $C \subseteq V$. Hence, by Proposition 4.5, $\sum_{U \in \mathcal{L}(A', B'|V)} \Delta F(U) = \Delta F(V) = 0$. Since for all $U \in \mathcal{L}(A', B'|V)$ with $U \neq V$, we have by Proposition 3.5 that $V \subset U \in \mathcal{L}(A, B|C)$ and, thus, $\Delta F(U) = 0$ by the induction hypothesis.

We now prove that statement (2) implies statement (3). Let $\mathcal{C}$ be a set of CI statements, let c be a CI statement, and assume that $\mathcal{C} \vdash c$. Since $\mathcal{F}$ has the zero-density property, we have that for each $F \in \mathcal{F}$, if $\models^a_F \mathcal{C}$ then for each $U \in \mathcal{L}(\mathcal{C})$, $\Delta F(U) = 0$. From $\mathcal{C} \vdash c$ and Proposition 3.5, we have $\mathcal{L}(C) \supseteq \mathcal{L}(c)$. Hence, for all $F \in \mathcal{F}$ we have that if F *a-satisfies* every CI statements in $\mathcal{C}$, then F *a-satisfies* c. Thus, $\mathcal{C} \models^a_{\mathcal{F}} c$.

Finally, statement (1) follows trivially from (3). □

5.2 Completeness

As with soundness in Subsection 5.1, we begin with the definition of the notion of *completeness* of inference system $\mathcal{A}$ for a given class of real-valued functions.

Definition 5.4 (Completeness)**.** Let $\mathcal{F}$ be a class of real-valued functions. We say that $\mathcal{A}$ is *complete* for the additive implication problem for CI statements relative to $\mathcal{F}$ if, for each set $\mathcal{C}$ of CI statements and each CI statement c, one has that $\mathcal{C} \models^a_{\mathcal{F}} c$ implies $\mathcal{C} \vdash c$.

We now introduce certain special real-valued functions that are at the basis of defining a property guaranteeing completeness of system $\mathcal{A}$.

Definition 5.5. Let $V \subseteq S$. The *Kronecker-density function* of V, denoted δ_V, is the real-valued function such that $\delta_V(V) = 1$ and $\delta_V(X) = 0$ if $X \neq V$. The *Kronecker-induced function* of V, denoted F_V, is the real-valued function whose density function is the Kronecker density function of V, i.e., for each $X \subseteq S$, $F_V(X) = \sum_{X \subseteq U \subseteq S} \delta_V(U)$, for each $X \subseteq S$.

We can now define a property on classes of real-valued functions that we will show to guarantee the completeness of system $\mathcal{A}$ for the additive implication problem.

Definition 5.6 (Kronecker property)**.** Let $\mathcal{F}$ be a class of real-valued functions, and let $\Omega \subseteq 2^S$. We say that $\mathcal{F}$ has the *Kronecker property* on Ω if, for each $U \in \Omega$, there exists a $c_U \in \mathbf{R}$ ($c_U \neq 0$), and a set $D_U = \{d_V \in \mathbf{R} \mid V \notin \Omega\}$ such that the following real-valued function is in $\mathcal{F}$:

$$F_{\Omega, c_U, D_U} := c_U F_U + \sum_{\substack{V \subseteq S \\ V \notin \Omega}} d_V F_V.$$

Note that for all $X \in \Omega$, $\Delta F_{\Omega, c_U, D_U}(X) = c_U$ if $X = U$ and $\Delta F_{\Omega, c_U, D_U}(X) = 0$ if $X \neq U$.

Let $\Omega^{(2)}$ be the set of all subsets of S that lack at least two of their elements, i.e., $\Omega^{(2)} = \{V \subset S \mid |V| \leq |S| - 2\}$. We can now prove that the Kronecker property on $\Omega^{(2)}$ implies the completeness of system $\mathcal{A}$.

Theorem 5.7. *Let $\mathcal{F}$ be a class of real-valued functions. If $\mathcal{F}$ has the Kronecker property on $\Omega^{(2)}$, then system $\mathcal{A}$ is complete for the additive implication problem for CI statements relative to $\mathcal{F}$.*

Proof. Assume that $\mathcal{F}$ has the Kronecker property on $\Omega^{(2)}$ but that $\mathcal{A}$ is not complete. Then there exists a set $\mathcal{C}$ of CI statements and a CI statement c such that $\mathcal{C} \models^a_{\mathcal{F}} c$ but $\mathcal{C} \not\vdash c$, or, equivalently by Theorem 3.8, $\mathcal{L}(c) \not\subseteq \mathcal{L}(\mathcal{C})$. Let $U \in \mathcal{L}(c) - \mathcal{L}(\mathcal{C})$. U must be an element in $\Omega^{(2)}$ by Lemma 3.3. Since $\mathcal{F}$ has the Kronecker property on $\Omega^{(2)}$, we know that there exists a $c_U \in \mathbf{R}$ ($c_U \neq 0$), and a set $D_U = \{d_V \in \mathbf{R} \mid V \notin \Omega^{(2)}\}$ such that $F_{\Omega^{(2)}, c_U, D_U} \in \mathcal{F}$. By Definition 5.6, $\Delta F_{\Omega^{(2)}, c_U, D_U}(X) = 0$ for all other $X \in \Omega^{(2)}$. From Proposition 4.5 it follows that $\models^a_{F_{\Omega^{(2)}, c_U, D_U}} \mathcal{C}$, but $\not\models^a_{F_{\Omega^{(2)}, c_U, D_U}} c$, a contradiction to $\mathcal{C} \models^a_{\mathcal{F}} c$. □

The following example demonstrates the zero-density and Kronecker properties.

Example 5.8. Let $S = \{a, b, c\}$, let $\mathcal{F}_1 = \{F_\emptyset, F_a, F_b, F_c\}$ and $\mathcal{F}_2 = \{F_x\}$, where the densities for each real-valued function are given by the table in Figure 2. The densities of the remaining subsets of S are assumed to be 0 for each function. Now, $\Omega^{(2)} = \{\emptyset, a, b, c\}$ and, therefore, $\mathcal{F}_1$ has the Kronecker property on $\Omega^{(2)}$ since $F_{\Omega^{(2)}, c_U, D_U} = F_U$ for all $U \in \Omega^{(2)}$, and the zero-density property. $\mathcal{F}_2$ does not have the Kronecker property. It also does not have the zero-density property as $\models^a_{F_x} I(b, c|\emptyset)$ but $\Delta F_x(\emptyset) \neq 0$.

	$\emptyset$	$\{a\}$	$\{b\}$	$\{c\}$
$\Delta F_\emptyset$	0.1	0	0	0
ΔF_a	0	-0.3	0	0
ΔF_b	0	0	-0.6	0
ΔF_c	0	0	0	0.9
ΔF_x	-0.2	0.2	0.6	0.3

Figure 2: Densities of several real-valued functions.

6 The Conditional Independence Implication Problem

While the theory presented so far has been concerned with the additive implication problem for CI statements, it is also applicable to the conditional independence implication problem. The link between these two problems is made with the concept of *multi-information functions* (Studený [9]) induced by probability measures. In this paper we will restrict ourselves to the class of *discrete* probability measures.

Definition 6.1. A *probability model* over $S = \{s_1, \ldots, s_n\}$ is a pair (dom, P), where dom is a domain mapping that maps each s_i to a finite domain $dom(s_i)$, and P is a probability measure having $dom(s_1) \times \cdots \times dom(s_n)$ as its sample space. For $A = \{a_1, \ldots, a_k\} \subseteq S$, we will say that $\mathbf{a}$ is a domain vector of A if $\mathbf{a} \in dom(a_1) \times \cdots \times dom(a_k)$.

In what follows, we will only refer to probability measures, keeping their probability models implicit.

Definition 6.2. Let $I(A, B|C)$ be a CI statement, and let P be a probability measure. We say that P *m-satisfies* $I(A, B|C)$, and write $\models^m_P I(A, B|C)$, if for every domain vector $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ of A, B, and C, respectively, $P(\mathbf{c})P(\mathbf{a}, \mathbf{b}, \mathbf{c}) = P(\mathbf{a}, \mathbf{c})P(\mathbf{b}, \mathbf{c})$.

Relative to the notion of *m-satisfaction* we can now define the *probabilistic conditional independence implication problem.*

Definition 6.3 (Probabilistic conditional independence implication problem)**.** Let $\mathcal{C}$ be a set of CI statements, let c be a CI statement, and let $\mathcal{P}$ be the class of discrete probability measures. We say that $\mathcal{C}$ *m-implies* c relative to $\mathcal{P}$, and write $\mathcal{C} \models^m_{\mathcal{P}} c$, if each function $P \in \mathcal{P}$ that *m-satisfies* the CI statements in $\mathcal{C}$ also *m-satisfies* the CI statement c. The set $\{c \mid \mathcal{C} \models^m_{\mathcal{P}} c\}$ will be denoted by $\mathcal{C}^*$.

Next, we define the *multi-information function* induced by a probability measure (Studený [9]), which is based on the Kullback-Leibler divergence (Kullback and Leibler [4]).

Definition 6.4. Let P and Q be two probability measures over a discrete sample space. Then, the relative entropy (Kullback-Leibler divergence) H is defined as

$$H(P|Q) := \sum_{\mathbf{x}} \{P(\mathbf{x}) \log \frac{P(\mathbf{x})}{Q(\mathbf{x})},\ P(\mathbf{x}) > 0\},$$

with $\mathbf{x}$ ranging over all elements of the discrete sample space.

Definition 6.5. Let P be a probability measure, and let H be the relative entropy. The *multi-information function* $M_P : 2^S \to [0, \infty]$ induced by P is defined as

$$M_P(A) := H(P^A | \prod_{a \in A} P^{\{a\}}),$$

for each non-empty subset A of S and $M_P(\emptyset) = 0$.[3]

The class of multi-information functions induced by the class of discrete probability measures $\mathcal{P}$ will be denoted by $\mathcal{M}$. We can now state the fundamental result of Studený that couples the probabilistic CI implication problem with the additive implication problem for CI statements relative to $\mathcal{M}$.

Theorem 6.6 (Studený [9])**.** *Let $\mathcal{C}$ be a set of CI statements and let c be a CI statement. Then, $\mathcal{C} \models^a_{\mathcal{M}} c$ if and only if $\mathcal{C} \models^m_{\mathcal{P}} c$.*

7 Saturated CI Statements - Soundness and Completeness of $\mathcal{A}$

In this section we show that system $\mathcal{A}$ is sound and complete for the probabilistic CI implication problem for *saturated* CI statements. We recall that a CI statement $I(A, B|C)$ is saturated if $ABC = S$. We begin by showing the following technical lemma.

[3]Here, P^A and $P^{\{a\}}$ denote the marginal probability measures of P over A and $\{a\}$, respectively.

Lemma 7.1. *The class of multi-information functions $\mathcal{M}$ induced by the class of discrete probability measures has the zero-density property with respect to saturated CI statements.*

Proof. We have to show that for each saturated CI statements c, for each $M \in \mathcal{M}$, and for each $U \in \mathcal{L}(c)$, if $\models^a_M c$, then $\Delta M(U) = 0$. The semi-graphoid inference rules are sound relative to the class of probability measures. Hence, in particular, by Theorem 6.6, *weak union* is sound relative to $\mathcal{M}$, i.e., $\{I(AD, B|C)\} \models^a_{\mathcal{M}} I(A, B|CD)$. Let $M \in \mathcal{M}$, let ΔM be the corresponding density function, and let $\models^a_M I(A, B|C)$ with $ABC = S$. In addition, let $I(A, B|C)$ be non-trivial since the proposition is obviously true for trivial CI statements. We will prove by downward induction on the semi-lattice $\mathcal{L}(A, B|C)$ that $\Delta M(U) = 0$ for each $U \in \mathcal{L}(A, B|C)$. Note that this proof is similar to the proof of Proposition 5.3. (Here, *weak union* is used instead of *decomposition* and *strong union*).

For the base case, we show for each $W \in \mathcal{W}(A, B|C)$ that $\Delta M(\overline{W}) = 0$. Let $W = \{a, b\}$. By repeatedly applying *weak union* we can derive $\models^a_M I(a, b|\overline{W})$ because $ABC = S$. Now, since $\mathcal{L}(a, b|\overline{W}) = \{\overline{W}\}$ we can conclude that $\Delta M(\overline{W}) = 0$.

For the induction step, let $V \in \mathcal{L}(A, B|C)$. The induction hypothesis states that $\Delta M(U) = 0$ for each $U \in \mathcal{L}(A, B|C)$ with U a strict superset of V. From the given CI statement $I(A, B|C)$ we can derive, again by *weak union*, $I(A', B'|V)$ with $VA'B' = S$ since $V - C \subseteq AB$. Since $\mathcal{L}(A', B'|V)$ contains only V and strict supersets V' of V, with $V' \in \mathcal{L}(A, B|C)$, we can conclude that $\sum_{U \in \mathcal{L}(A', B'|V)} \Delta F(U) = \Delta F(V) = 0$ by the induction hypothesis. □

We are now in the position to prove that inference system $\mathcal{A}$ is sound and complete for the probabilistic implication problem for *saturated* conditional independence statements.

Theorem 7.2. *$\mathcal{A}$ is sound and complete for the probabilistic conditional independence implication problem for saturated CI statements.*

Proof. The soundness follows directly from Lemma 7.1, Theorem 5.3, and Theorem 6.6. To show completeness, notice that the semi-graphoid axioms are derivable under inference system $\mathcal{A}$. Furthermore, Geiger and Pearl proved that the semi-graphoid axioms are complete for the probabilistic conditional independence implication problem for *saturated* CI statements (Geiger and Pearl [3]). □

8 CI Statements - Completeness of $\mathcal{A}$

In this section we will show that inference system $\mathcal{A}$ is complete for the probabilistic conditional independence implication problem. We first prove that $\mathcal{M}$ has the Kronecker property on $\Omega^{(2)}$. To show this, it would be sufficient to construct a set of discrete probability measures whose induced multi-information functions are Kronecker-induced functions. However, instead of taking this route, we pursue a different approach by first focusing on results with respect to *saturated* CI statements. We first need the following simple lemma.

Lemma 8.1. *For $U \subseteq S$, $\{X \in \Omega^{(2)} \mid X \supseteq U\} = \bigcup_{\substack{U_1 \cup U_2 = \overline{U} \\ U_1 \cap U_2 = \emptyset}} \mathcal{L}(U_1, U_2|U)$.*

Proposition 8.2. *Let $\mathcal{F}$ be a class of real-valued functions. If $\mathcal{A}$ is sound and complete for the additive implication problem relative to $\mathcal{F}$ for* saturated *CI statements, then $\mathcal{F}$ has the Kronecker property on $\Omega^{(2)}$.*

Proof. If $|S| \leq 1$, then $\Omega^{(2)} = \emptyset$ and the statement follows trivially. Hence, assume that $|S| \geq 2$. Suppose that $\mathcal{A}$ is sound and complete for saturated CI statements but that $\mathcal{F}$ does not have the Kronecker property on $\Omega^{(2)}$. Then there exists a set $U \in \Omega^{(2)}$ such that for each $c_U \in \mathbf{R}$ ($c_U \neq 0$), and for each set $D_U = \{d_V \in \mathbf{R} \mid V \notin \Omega^{(2)}\}$ we have $F_{\Omega^{(2)}, c_U, D_U} \notin \mathcal{F}$. Now, let $\mathcal{C}$ be the set of saturated CI statements

$$\{\ I(U, \overline{U}|\emptyset)\ \} \cup \bigcup_{\substack{U_1 \cup U_2 = U \\ U_1 \cap U_2 = \emptyset}} \{\ I(U_1, U_2|\overline{U})\ \} \cup$$

$$\bigcup_{v \in \overline{U}} \bigcup_{\substack{V_1 \cup V_2 = \overline{U} - \{v\} \\ V_1 \cap V_2 = \emptyset}} \{\ I(V_1, V_2|U \cup \{v\})\ \},$$

and let c be the saturated CI statement $I(U_1, U_2|U)$ for some non-empty sets U_1 and U_2. Notice that such sets exist because $|\overline{U}| \geq 2$. By Lemma 8.1 it is $\mathcal{L}(\mathcal{C}) = \Omega^{(2)} - \{U\}$ and $U \in \mathcal{L}(c) \subseteq \Omega^{(2)}$ and therefore $\mathcal{L}(c) \nsubseteq \mathcal{L}(\mathcal{C})$. Hence, by Theorem 3.8, $\mathcal{C} \nvdash c$. We now show that $\mathcal{C} \models^a_{\mathcal{F}} c$ to obtain the contradiction to the completeness of $\mathcal{A}$. If there does not exist an $F \in \mathcal{F}$ which *a-satisfies* $\mathcal{C}$ we are done because then $\mathcal{C} \models^a_{\mathcal{F}} c$ follows trivially. Thus, let F be in $\mathcal{F}$ and assume that $\models^a_F \mathcal{C}$. Since $\mathcal{A}$ is sound relative to $\mathcal{F}$ for saturated CI statements, we know by Theorem 5.3 that $\mathcal{F}$ has the zero-density property. Thus, $\Delta F(X) = 0$ for each $X \in \Omega^{(2)}$ with $X \neq U$. But then $\Delta F(U) = 0$ since otherwise there would exist a $c_U \in \mathbf{R}$, $c_U = \Delta F(U) \neq 0$, and a set $D_U = \{d_V \in \mathbf{R} \mid V \notin \Omega^{(2)}\}$ such that $F_{\Omega^{(2)}, c_U, D_U} = F \in \mathcal{F}$. Hence, F must be a function whose density is zero on every element of $\Omega^{(2)}$. Thus, $\models^a_F c$ and it follows that $\mathcal{C} \models^a_{\mathcal{F}} c$. □

The completeness of $\mathcal{A}$ for the CI implication problem can now be proved based on the previous results.

Theorem 8.3. *$\mathcal{A}$ is complete for the probabilistic conditional independence implication problem.*

Proof. We know from Theorem 7.2 that $\mathcal{A}$ is sound and complete relative to $\mathcal{M}$ for saturated CI statements. Now, by Proposition 8.2, $\mathcal{M}$ has the Kronecker property on $\Omega^{(2)}$. Finally, through Theorem 5.7 and Theorem 6.6, the statement follows. □

Example 8.4. (Studený [9]) described the following sound inference rule relative to discrete probability measures which refuted the conjecture (Pearl [6]) that the semi-graphoid axioms are complete for the probabilistic CI implication problem:

$I(A, B|CD) \wedge I(C, D|A) \wedge I(C, D|B) \wedge I(A, B|\emptyset) \rightarrow I(C, D|AB) \wedge I(A, B|C) \wedge I(A, B|D) \wedge I(C, D|\emptyset)$.

By applying *strong contraction* to the statements $I(A, B|\emptyset), I(C, D|A)$, and $I(C, D|B)$ we can derive the statement $I(C, D|\emptyset)$. All the other statements can be derived using *strong union*.

Remark 8.5. The inference system $\mathcal{A}$ without *strong contraction* is *not* complete. The consequence $I(C, D|\emptyset)$ of the clause from Example 8.4 cannot be derived from the antecedents without *strong contraction*.

9 Complete Axiomatization of Stable Independence

When new information is available to a probabilistic system the set of associated relevant CI statements changes dynamically. However, some of the CI statements will continue to hold. These CI statements were termed *stable* by de Waal and van der Gaag [2]. A first investigation of their structural properties was undertaken by Matúš who used the term *ascending* conditional independence (Matúš [5]). Every set of CI statements can be partitioned into its *stable* and *unstable* part. We will show that inference system $\mathcal{A}$ is sound and complete for the probabilistic CI implication problem for *stable* conditional independence statements.

Definition 9.1. Let $\mathcal{C}$ be a set of CI statements, and let $\mathcal{C}^{SG+}$ be the semi-graphoid closure of $\mathcal{C}$. Then $I(A, B|C)$ is said to be *stable* in $\mathcal{C}$, if $I(A, B|C') \in \mathcal{C}^{SG+}$ for all sets C' with $C \subseteq C' \subseteq S$.

Theorem 9.2. *Let $\mathcal{C}_S$ be a set of stable CI statements. Then, $\mathcal{A}$ is sound and complete for the probabilistic conditional independence implication problem for $\mathcal{C}_S$, or, equivalently, $\mathcal{C}^*_S = \mathcal{C}^+_S$.*

Proof. The soundness follows from Theorem 5.3 and from *strong union* and *decomposition* being sound inference rules relative to $\mathcal{M}$ for stable CI statements. The completeness follows from Theorem 8.3. □

Remark 9.3. The previous result is also interesting with respect to the problem of finding a minimal, non-redundant representation of stable independence relations. Here, lattice-inclusion could aid the lossless compaction of representations of stable CI statements: $\mathcal{L}(\mathcal{C}_S - \{c\}) = \mathcal{L}(\mathcal{C}_S)$ if and only if c is redundant in $\mathcal{C}_S$.

10 Falsification Algorithm

Theorem 3.8 and Theorem 8.3 lend themselves to a *falsification algorithm*, that is, an algorithm which can falsify instances of the probabilistic conditional independence implication problem. We consider the following corollary which directly follows from these two results.

Corollary 10.1. *Let $\mathcal{C}$ be a set of CI statements, and let $\mathcal{P}$ be the class of discrete probability measures. If $\mathcal{L}(\mathcal{C}) \not\supseteq \mathcal{L}(c)$, then $\mathcal{C} \not\models^m_{\mathcal{P}} c$.*

If the falsified implications were, on average, only a small fraction of all those that are falsifiable, the result would be disappointing from a practical point of view. Fortunately, we will not only be able to show that a large number of implications can be falsified by the "lattice-exclusion" criterion identified in Corollary 10.1, but also that polynomial time heuristics exist that provide good approximations of said criterion.

Falsification Criterion. `Input`: A set of CI statements $\mathcal{C}$ and a CI statement c. `Test`: if $\mathcal{L}(\mathcal{C}) \not\supseteq \mathcal{L}(c)$, return "false", else return "unknown."

Heuristic 1. `Input`: A set of CI statements $\mathcal{C}$ and a CI statement $I(A, B|C)$. `Test`: if for each $I(A', B'|C') \in \mathcal{C}$ it is $C \not\supseteq C'$, return "false", else return "unknown."

Heuristic 2. `Input`: A set of CI statements $\mathcal{C}$, and a CI statement $I(A, B|C)$. `Test`: if there exists one $W \in \mathcal{W}(A, B|C)$ such that for all $I(A', B'|C') \in \mathcal{C}$ it is $W \notin \mathcal{W}(A', B'|C')$, return "false", else return "unknown."

It follows from Lemma 3.3 that if one of the two heuristics returns "false," then $\mathcal{L}(\mathcal{C}) \not\supseteq \mathcal{L}(c)$, and therefore $\mathcal{C} \not\models^m_{\mathcal{P}} c$ by Corollary 10.1.

Example 10.2. Let S be a finite set, and $A, B, C,$ and D be pairwise disjoint subsets of S. The inference rule *intersection*, $I(A, B|DC) \wedge I(A, D|BC) \rightarrow I(A, BD|C)$, is *not* sound relative to the class of discrete probability measures. Heuristic 1 can reject this instance of the implication problem in polynomial time in the size of S.

Remark 10.3. The falsification criterion leads in fact to a *family* of polynomial time heuristics. While Heuristic 1 checks if the unique *meet* (greatest lower bound) of the semi-lattice $\mathcal{L}(c)$ is not in $\mathcal{L}(\mathcal{C})$ and Heuristic 2 if the (potentially multiple) *joins* (least upper bounds) of the semi-lattice $\mathcal{L}(c)$ are not in $\mathcal{L}(\mathcal{C})$, we may select additional elements in the semi-lattice $\mathcal{L}(c)$ that are located between these two extrema to derive more falsification heuristics.

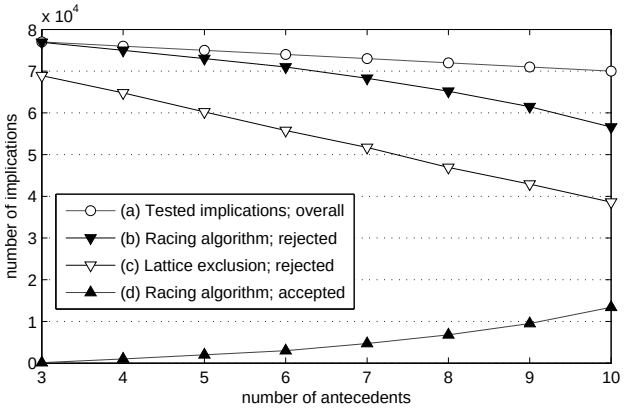


Figure 3: Rejection and acceptance curves of the racing and falsification algorithms, respectively, for five attributes.

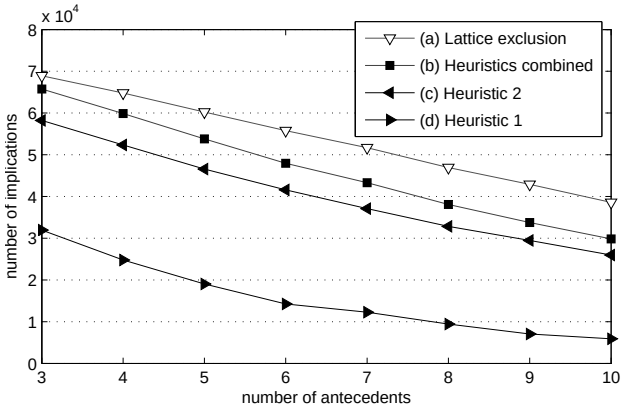


Figure 4: Falsifications based on the lattice-exclusion criterion and the heuristics, for five attributes. The combination of the heuristics reaches 95% of the falsifications of the full-blown lattice exclusion criterion for 3 antecedents down to 77% for 10 antecedents.

With our experiments we want to show that (1) the lattice-exclusion criterion can falsify a large fraction of all falsifiable implications, and (2) that the two provided heuristics are good approximation of the full-blown lattice-exclusion criterion. To make our outcomes comparable to existing results, we adopted the experimental setup for the *racing algorithm* from Bouckaert and Studený [1] (also using 5 attributes). A thousand sets of antecedents each were generated by randomly selecting 3 up to 10 elementary CI statements, resulting in a total of 8000 sets of antecedents.[4]

[4] An elementary CI statement is of the form $I(a, b|C)$, where $a, b \in S$ and $C \subseteq S - \{a, b\}$.

The *falsification algorithm* and the heuristics were run on these sets with each of the remaining elementary CI statements as consequence, one at a time. Since there are 80 elementary CI statements for 5 attributes, this resulted in 77000 implication problems for sets with 3 antecedents, 76000 for sets with 4 antecedents, down to 70000 for sets with 10 antecedents.

The rejection procedure of the *racing algorithm* is rooted in the theory of imsets: an instance is rejected if one of the supermodular functions constructed by the algorithm is a counter-model for this instance. It has exponential running time and might reject implications that actually *do* hold. This is a consequence of the fact that $\mathcal{M}$ is a *strict* subset of the class of all supermodular functions. (See Examples 4.1 and 6.2 in Studený's monograph [9].) The *falsification algorithm* based on Corollary 10.1, on the other hand, ensures that if an instance of the implication problem is rejected, then it is guaranteed not to be valid.

Figure 3 shows the rejection curves of the *racing algorithm* (b) and the *falsification algorithm* (c), respectively, and the acceptance curve of the racing algorithm (d). The area between the two rejection curves can be interpreted as the "decision gap", i.e., the amount of instances of the implication problem for which the validity is unknown. The curve marked with circles (a) depicts the total number of tested instances. Figure 4 depicts the rejection curves for the *falsification algorithm* (a), for the combination of Heuristic 1 and Heuristics 2 (b), and for Heuristic 2 (c) and Heuristic 1 (d) run separately. The combination of the heuristics compares favorable with the full-blown *falsification criterion*. The experiments also show that Heuristic 2 is more effective than Heuristic 1.

11 Conclusion and Future Work

A complete inference system for the probabilistic conditional independence implication problem was presented and related to the lattice-exclusion criterion. We derived polynomial time approximations that can be used as a preprocessing step to efficiently shrink the search space of possibly valid inferences. We already have experimental evidence that our approach scales to much larger instances of the implication problem than those reported on in this paper. This could, for instance, provide insights into combinatorial bounds for the number of (stable) CI structures. The falsification algorithm and the heuristics can be combined with algorithms that infer valid implications, like the one based on structural imsets which is used as part of the *racing algorithm* [1]. In addition, the lattice exclusion criterion and the heuristics can be utilized to store information about conditional independencies more efficiently, using non-redundant representations. Overall, we believe that the lattice-theoretic framework for reasoning about conditional independence is a novel and powerful tool. We conjecture that there are interesting connections between our theory and Studený's theory of imsets which we will continue to investigate.

Acknowledgments

We thank Remco Bouckaert for providing us with the source code of the *racing algorithm*. We also want to thank Milan Studený for the information he provided us concerning the complete axiomatization of the implication problem for four attributes, František Matúš for helpful feedback on an earlier draft, and the anonymous reviewers whose comments helped to improve the quality of the paper.

References

[1] R. R. Bouckaert and M. Studený. Racing algorithms for conditional independence inference. *Int. J. Approx. Reasoning*, 45(2):386–401, 2007.

[2] P. de Waal and L. C. van der Gaag. Stable independence and complexity of representation. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 112–119, 2004.

[3] D. Geiger and J. Pearl. Logical and algorithmic properties of conditional independence and graphical models. *The Annals of Statistics*, 21(4):2001–2021, 1993.

[4] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[5] F. Matúš. Ascending and descending conditional independence relations. In *Transactions of the 11th Prague Conference on Information Theory*, pages 189–200, 1992.

[6] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.

[7] B. Sayrafi and D. V. Gucht. Differential constraints. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 348–357, 2005.

[8] M. Studený. Conditional independence statements have no complete characterization. In *Transactions of the 11th Prague Conference on Information Theory*, pages 377–396, 1992.

[9] M. Studený. *Probabilistic Conditional Independence Structures*. Springer-Verlag, 2005.

Learning Hidden Markov Models for Regression using Path Aggregation

Keith Noto
Dept. of Computer Science and Engineering
University of California San Diego
La Jolla, CA 92093
knoto@cs.ucsd.edu

Mark Craven
Dept. of Biostatistics and Medical Informatics
University of Wisconsin
Madison, WI 53706
craven@biostat.wisc.edu

Abstract

We consider the task of learning mappings from sequential data to real-valued responses. We present and evaluate an approach to learning a type of hidden Markov model (HMM) for regression. The learning process involves inferring the structure and parameters of a conventional HMM, while simultaneously learning a regression model that maps features that characterize paths through the model to continuous responses. Our results, in both synthetic and biological domains, demonstrate the value of jointly learning the two components of our approach.

1 Introduction

A wide array of problems in speech and language processing, biology, vision, and other application domains involve learning models that map sequences of inputs into some type of output. Common types of task include learning models that classify sequences (*e.g.*, [12]), segment or parse them (*e.g.*, [13]), or map input sequences to output sequences (*e.g.*, [3]). Here we consider the task of learning models that map input sequences to real-valued responses. We present an approach to this problem that involves simultaneously learning a hidden Markov model (HMM) and a function that maps paths through this model to real-valued responses. We evaluate our approach using synthetic data sets and a large collection from a yeast genomics study.

The type of task that we consider is illustrated in Figure 1. This is a type of regression task in that the learner must induce a mapping from a given input sequence to a real-valued response. In particular, we assume that the real-valued responses can be represented as a function of the presence and arrangement of particular *motifs* that occur in the sequences. Each of these motifs is a pattern that allows some variability in the subsequences that match it.

We assume that neither these motifs nor their locations in the sequences are given to the learner, but instead must be discovered during the learning process. The learner must also determine the extent to which each motif and its relationships to other motifs contribute to the response variable.

Our research is motivated by a class of problems in computational biology that involve inferring the extent to which particular properties of genomic sequences determine certain responses in a cell. For example, the level at which an individual gene is expressed in a given condition often depends on the presence of particular *activators* that bind to sequence motifs nearby the gene. Moreover, the number of binding motifs, their arrangement in the sequence, and intrinsic properties of the motifs themselves may contribute to the response level of the gene. Thus, in order to explain the expression levels of genes in some condition, a model needs to be able to map these sequence properties into continuous values.

The approach that we present involves simultaneously learning (i) the structure and parameters of a hidden Markov model, and (ii) a function that maps paths through the model to real-valued responses. The hidden Markov model is able to represent the relevant sequence motifs and the regression model is able to represent the mapping from occurrences of these motifs to the response variable.

There are several bodies of related research. First, there is a wide variety of architectures and methods for learning HMMs [19], stochastic context-free grammars [17], and related probabilistic sequence models, such as conditional random fields [13]. For some types of problems, these models include continuous random variables. Typically these continuous variables depend only on a few other variables, and the dependencies are encoded at the outset. In our models, in contrast, the

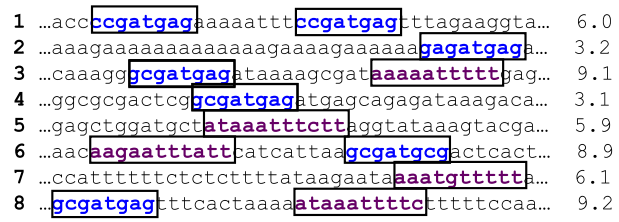


Figure 1: An example of the sequence-based regression task that we are addressing. Each row in the figure represents a particular training instance. Each instance consists of a DNA sequence and an associated real-valued output. The sequences in this example contain two types of motifs; m_1 whose consensus sequence is gcgatgag and m_2 whose consensus sequence is aaaaattttt. In the tasks we consider, the motifs and their occurrences are hidden. The learning task involves discovering the motifs and their occurrences in the sequences, and inferring a function that maps motif occurrences to the real value associated with each sequence. In this example, $y \approx 3 \times v_1 + 6 \times v_2$, where v_1 represents the number of occurrences of m_1 and v_2 represents the number of occurrences of m_2.

continuous response variable may depend on quite a few variables that characterize the input sequence, and these variables and their dependencies are determined during the learning process.

There is also large corpus of work on the topic of regression methods [7]. Most regression methods assume that each instance is represented using a fixed-size set of pre-defined variables. Our approach, on the other hand, assumes that each instance is represented by a sequence of values, but these sequences may vary in their lengths and the positions of the relevant sequence elements may vary as well. Moreover, our method is designed to derive a set of variables, from the given sequences, that are predictive of the response variable.

Various groups have devised kernels defined over sequences that provide mappings from sequence features to real numbers. These string kernels can be used to map sequences to feature vectors which can then be used for regression or classification [15]. However, these kernels encode predefined sequence features. In contrast, our method is designed to learn which sequence features best provide input to the regression part of the model. Jaakkola et al. [8] have used HMMs to identify the relevant aspects of sequences then, in a second step, the Fisher kernel for classification based on the HMM representations. Our experiments in Section 3 indicate that more accurate models can be learned by using the training signal to guide the discovery of relevant sequence features.

Several inductive logic programming (ILP) methods for learning regression models have been previously developed [9, 11]. The algorithms are similar to ours in that they can handle variable-sized descriptions of instances and they employ an expressive representation for regression tasks. They differ from our approach in that they are not designed to discover sequence motifs and use properties of these motifs in the regression model. This aspect of our approach is essential for the problems we consider.

A variety of methods have been developed for discovering motifs in biological sequences [1, 14, 16], and for identifying arrangements of motifs that are involved in particular biological processes [20, 22]. These methods are designed for either unsupervised pattern discovery or supervised classification tasks. They either try to find motifs that are over-represented in a given set of sequences, or they try to find motif arrangements that distinguish two given sets of sequences. Our method, in contrast, is intended for regression tasks. There are also several methods that learn models that characterize gene-expression responses in terms of sequence features. Some of these approaches first group expression values into discrete sets and then frame the problem as a classification task [2, 21]. Other methods use a two-phase approach that first identifies candidate motifs without the use of expression data, and then learns a regression model from expression data and these fixed sequence features [5, 23]. The key difference between our approach and these methods is that, in our approach, the real-valued response associated with each sequence is a training signal that has a direct influence on the sequence features represented by the model.

2 Approach

The task that we consider is to learn a function which maps a given discrete character sequence $\mathbf{x} = \{x_1, x_2, ..., x_L\}$ to a real-valued scalar y. In this section we describe the representation we employ and discuss the procedure we use for learning the models.

2.1 Representation

We assume that there are certain features, or motifs, present in each sequence $\mathbf{x}$ that determine the associated y value. However, in the tasks that we consider, the learner is given only sequences and their response values, and must discover both the motifs and their locations in the sequences. Thus, our approach involves learning the structure and parameters of a hidden Markov model that represents these motifs. The other key component of our learned model is a regression function that maps from occurrences of the motifs to y values. In particular, we associate certain states in the HMM with motifs, and represent the putative occurrences of motifs in a given sequence by keeping track of the number of times that each of these states is visited. That is, a subset of the states in the HMM

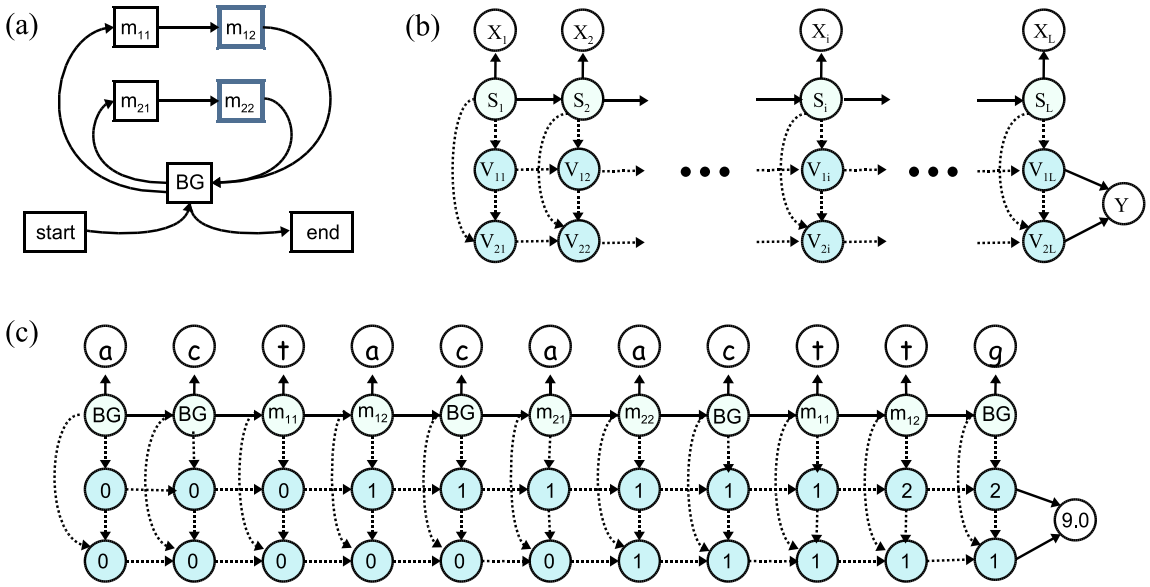


Figure 2: An HMM for regression and the corresponding graphical model. Panel **(a)** shows the state topology of a simple HMM that is able to represent occurrences of two types of motifs in given sequences. Each motif consists of exactly two DNA bases. For the ith motif, these bases are emitted by the states m_{i1} and m_{i2}. The state labeled BG emits the remaining "background" of the sequence. To calculate the distribution over the possible motif occurrences for each sequence, we count visits to states m_{12} and m_{22}. Panel **(b)** shows the structure of the corresponding graphical model when processing a sequence of length L. The X_i variables represent the observable sequence characters. The S_i variables represent the corresponding HMM state for each position in the input sequence. The V_{1i} (V_{2i}) variables represent the number of visits to state m_{12} (m_{22}) at or before the ith character in the input sequence. The Y variable represents the real-valued response for the given instance. Probabilistic dependencies are illustrated using solid lines and deterministic dependencies are illustrated using dashed lines. Panel **(c)** shows the instantiation of variables in the graphical model for the instance (actacaacttg, 9.0) and a particular path through the HMM that visits m_{12} twice and m_{22} once.

$\{c_1, c_2, ..., c_N\}$ are designated as "counted" states, and a path through the model defines an integer vector $\mathbf{v} = \langle v_1, v_2, ...v_N \rangle$, where each v_k is the number of visits to state c_k.

More generally, we have uncertainty about the "correct" path through the model, and therefore uncertainty about the number of visits to each state c_k. Consider the HMM shown in Figure 2(a). There are two types of motifs, each two characters long. The motif occurrences are assumed to be interspersed with variable-length "background" sequence which is modeled by the BG state[1]. In this HMM, we count visits to each motif (*i.e.*, $c_1 = \mathrm{m}_{12}$, $c_2 = \mathrm{m}_{22}$). Figure 2(b) shows the corresponding graphical model when processing a sequence of length L. Each circle represents a random variable in the model, and edges represent direct dependencies. Probabilistic dependencies are shown with solid lines and deterministic dependencies are shown with dashed lines. Figure 2(c) shows the values taken on by the variables in the model for a case in which $\mathbf{x}$ = actacaacttg, $y = 9.0$, and we have assumed a particular path through the HMM. This path involves going through the top motif twice and the lower motif once. We discuss each of the random variables in turn.

Each variable X_i represents the ith character in the input sequence $\mathbf{x}$. The variable S_i represents the HMM state that we are in after explaining the first i characters of $\mathbf{x}$. This state depends directly on the previous state, and this dependency is encoded by the HMM transition parameters. The variable X_i depends on the corresponding state variable S_i, and this relationship is encoded via the HMM emission parameters. In the problems we consider, these state sequences are hidden during both training and testing.

Each $V_{k,i}$ represents the number of visits to state c_k in the paths through the HMM which explain the first i characters of $\mathbf{x}$. These variables are also hidden and depend on the HMM state S_i and the corresponding variable from the previous position, $V_{k,i-1}$. They are updated as follows:

$$P(V_{k,i} = v | s_i, V_{k,i-1}) = \begin{cases} P(V_{k,i-1} = v-1) & \text{if } s_i = c_k \\ P(V_{k,i-1} = v) & \text{otherwise} \end{cases} \quad (1)$$

Moreover, as illustrated by the edges between the bottom two nodes in each column of Figure 2(b), we may

[1] As an alternative to a self-transition, our background models include a probability distribution over lengths of subsequence, making these models *generalized* [4] or hidden *semi*-Markov models.

represent dependencies among the $V_{k,i}$ variables at the ith position. Doing this enables us to model an arbitrary joint distribution characterizing the visits to the "counted" states.

Finally, the variable Y in Figure 2(b) is the real-valued response associated with the sequence in question. Its value depends on the number of visits to all counted states after explaining the entire sequence. Thus, the last column of visit count variables in Figure 2(b) determines the response value, $y = f(\langle V_{1L}, ..., V_{NL}\rangle)$.

We represent Y using a linear Gaussian model. Let $\mathbf{V}$ denote the vector of variables $\langle V_{1L}, ..., V_{NL}\rangle$, and let $\mathbf{v}$ denote a particular vector of visit counts $\langle v_{1L}, ..., v_{NL}\rangle$. Given a specific $\mathbf{v}$, this model represents the probability distribution of Y as a Gaussian whose mean is a linear function of the visit-count vector:

$$p(Y|\mathbf{v}) \equiv N(\beta_1 v_{1L} + \beta_2 v_{2L} + ... + \beta_N v_{NL}, \sigma^2) \quad (2)$$

Here, each β_k is an unknown model parameter which represents the contribution to the response variable of each occurrence of the motif represented by state c_k. The standard deviation σ is also a model parameter to be learned.

Since the $\mathbf{V}$ variables are hidden, we may infer a distribution for Y given a sequence $\mathbf{x}$, by marginalizing out $\mathbf{V}$, based on its likelihood given $\mathbf{x}$ which is computed by our hidden Markov model.

$$p(Y|\mathbf{x}) = \sum_{\mathbf{v}} p(Y|\mathbf{v})P(\mathbf{v}|\mathbf{x}). \quad (3)$$

Note that (3) involves a summation over the possible values of $\mathbf{v}$. In general, the number of possible values can increase exponentially with the number of counted states N, but in practice, N is a small number of features. There are tasks in which it is useful to have a larger number of counted states, and in many such cases, the calculations are still tractable because the HMM topology and the sequence characters in $\mathbf{x}$ prevent otherwise possible values of $\mathbf{v}$ from having a nonzero probability. If the number of possible values of $\mathbf{v}$ is still prohibitively large, then we may choose to consider only its most likely values, or we may instead calculate the expected value of $\mathbf{V}$ which is $\hat{\mathbf{v}} = \langle \hat{v}_1, \hat{v}_2, ..., \hat{v}_N\rangle$, where each $\hat{v}_k = \sum_\pi P(\delta(\pi)|\mathbf{x})$, where $\delta(\pi)$ deterministically maps a path π through the HMM to the appropriate vector. $\hat{\mathbf{v}}$ is computed efficiently using dynamic programming. $p(Y|\mathbf{x})$ is then estimated as $p(Y|\hat{\mathbf{v}})$.

2.2 Parameter Learning

Given an HMM structure, we select parameter values to maximize the joint probability of the observed input sequences and their associated response values. Taking into account uncertainty in the "correct" path for each given sequence, and the dependencies represented in the model, we can express the objective function as:

$$\arg\max_{\boldsymbol{\Theta},\boldsymbol{\beta},\sigma} \prod_{(\mathbf{x},y)} \sum_{\mathbf{s}} [P(\mathbf{s}:\boldsymbol{\Theta})P(\mathbf{x}|\mathbf{s}:\boldsymbol{\Theta})p(y|\mathbf{v}=\delta(\mathbf{s}):\boldsymbol{\beta},\sigma)] \quad (4)$$

where s_i is the HMM state we are in at the ith character of $\mathbf{x}$. This product ranges over all of the $(\mathbf{x}, y)$ pairs in the training set, $\boldsymbol{\Theta}$ represents the usual set of HMM state transition and character emission parameters, and $\boldsymbol{\beta}$ and σ are the parameters of the regression model described above. Note that, because a given $\mathbf{s}$ maps deterministically onto a particular $\mathbf{v} = \delta(\mathbf{s})$, we do not need to sum over $\mathbf{v}$. We train using an expectation-maximization (EM) approach which is a slight modification of the standard Baum-Welch algorithm for HMMs [19].

E-step: EM algorithms are already widely used for training HMM models to represent sequence motifs (*e.g.* [1]). The difference between standard Baum-Welch and our approach is that we calculate the expected values for our hidden variables taking into account y as well as $\mathbf{x}$. To accomplish this, we calculate a probability distribution over $\mathbf{V} = \langle V_{1L}, V_{2L}, ..., V_{NL}\rangle$ given $\mathbf{x}$, y and our model parameters by considering each possible value for $\mathbf{v}$. The probability is given by

$$P(\mathbf{v}|\mathbf{x}, y:\boldsymbol{\Theta},\boldsymbol{\beta},\sigma) = \frac{1}{Z}p(y|\mathbf{v},\mathbf{x}:\boldsymbol{\beta},\sigma) \quad (5)$$

where Z is a normalization constant. This is the base-case initialization for the backward calculations involved in standard E-step of Baum-Welch. Apart from this, we compute the expected values for all hidden variables $V_{1,1}, V_{1,2}, ...V_{N,L-1}$ and $S_1, S_2, ..., S_L$ using the standard forward-backward calculations, which update the probability distribution over $V_{1L}, V_{2L}, ..., V_{NL}$ accounting for $\mathbf{x}$ and $\boldsymbol{\Theta}$.

M-step: Apart from the initialization of the backward calculations described above, the estimation of our HMM parameters $\boldsymbol{\Theta}$ are calculated using standard M-step of Baum-Welch. We choose our regression model parameters $\boldsymbol{\beta}$ and σ using standard least-squares regression, except that the possible values for $\mathbf{v}$ given a training example i are weighted by their likelihood given $\mathbf{x}_i$ and y_i. Thus, we minimize the total *expected* squared difference between the observed and predicted response values in the training set. This is calculated by marginalizing over the possible values for $\mathbf{v}$, according the their likelihood $P(\mathbf{v}|\mathbf{x}_i, y_i:\Theta,\boldsymbol{\beta},\sigma)$, which is calculated in the E-step:

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \sum_{i}^{D} \sum_{\mathbf{v}} P(\mathbf{v}|\mathbf{x}_i, y_i:\boldsymbol{\Theta},\boldsymbol{\beta},\sigma)(y_i - \boldsymbol{\beta}\cdot\mathbf{v})^2 \quad (6)$$

where D is the training set size. Let H_k be the maximum number of visits to state c_k. [2] $\mathcal{V} = \prod_k^N H_k$ is the number of possible values of $\mathbf{v}$. Equation (6) has the closed-form solution:

$$\hat{\boldsymbol{\beta}} = (\mathbf{A}^T \boldsymbol{\Gamma} \mathbf{A})^{-1} \mathbf{A}^T \boldsymbol{\Gamma} \mathbf{b} \tag{7}$$

where $\mathbf{A}$ is a $\mathcal{V}D \times N$ matrix of all possible values of $\mathbf{v}$ for each training example, $\mathbf{b}$ is a $\mathcal{V}D \times 1$ vector of the y response values corresponding to each row of $\mathbf{A}$, and $\boldsymbol{\Gamma}$ is a $\mathcal{V}D \times \mathcal{V}D$ diagonal matrix, where each value γ_i represents the likelihood of $\mathbf{v}$ in row i of $\mathbf{A}$, given the appropriate training example, *i.e.* $\gamma_1 = P(\mathbf{v} = \langle 0, 0, ..., 0 \rangle | \mathbf{x_1}, y_1 : \boldsymbol{\Theta}, \boldsymbol{\beta}, \sigma)$.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ & \vdots & & \\ H_1 & H_2 & \cdots & H_N \\ 0 & 0 & \cdots & 0 \\ & \vdots & & \\ H_1 & H_2 & \cdots & H_N \end{bmatrix}, \mathbf{b} = \begin{bmatrix} y_1 \\ \vdots \\ y_1 \\ y_2 \\ \vdots \\ y_D \end{bmatrix},$$

$$\boldsymbol{\Gamma} = \begin{bmatrix} \gamma_1 & 0 & 0 & \cdots & 0 \\ 0 & \gamma_2 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ & & & \ddots & 0 \\ 0 & 0 & \cdots & 0 & \gamma_{\mathcal{V}D} \end{bmatrix}. \tag{8}$$

The value for σ is estimated from the (minimized) expected difference between our best fit line and the data points.

Again, if the number of possible values of $\mathbf{v}$ is prohibitively large, we can sample from the distribution of $\mathbf{v}$ or we can use the expected number of visits $\hat{v}_k$ to each c_k, and solve $\hat{\boldsymbol{\beta}} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$, where

$$\mathbf{A} = \begin{bmatrix} \hat{\mathbf{v}_1} \\ \hat{\mathbf{v}_2} \\ \vdots \\ \hat{\mathbf{v}_D} \end{bmatrix} \mathbf{b} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_D \end{bmatrix} \tag{9}$$

and $\hat{\mathbf{v}}_i$ is the vector of expected visits calculated for the ith training sequence, $\mathbf{x}_i$.

2.3 Structure Learning

Our task includes learning the underlying model structure as well as parameters. This structure refers to the set of states and transitions that define the HMM topology. Although our regression approach applies to arbitrary HMM structures, we are primarily interested in the occurrence and arrangement of motifs. These motifs represent classes of short substrings with character preferences at each position. Whereas Figure 2(a) shows a model that can represent an arbitrary number of occurrences of two very short motifs, more sophisticated arrangements can be encoded in the HMM topology. Consider the structures shown in Figure 3. Here each shaded, rounded shape represents a sequence of states modeling a single motif. This model considers not only the presence of particular motifs, but also their logical arrangement. The counted states (although they may be visited at most once) correspond to each combination of these substrings, and thus the response variable Y is a function of not only the presence of particular sequence characters, but it is also senstitive to whether or not they appear in the preferred order.

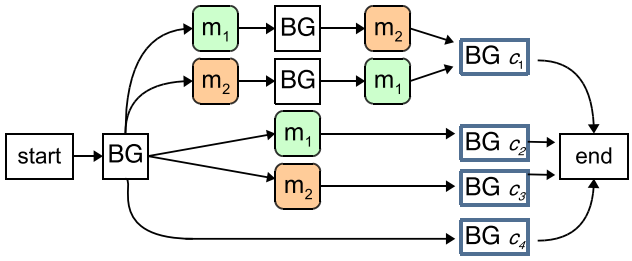


Figure 3: An HMM structure that considers both the presence and arrangement of over-represented sequence characters. This example has two such substrings, m_1 and m_2, which are chains of states but encapsulated here as single states. Other characters are explained by the BG background states. We count visits to states labeled c_x.

Instead of searching through the space of arbitrary HMM topologies by adding and removing individual states and transitions, our search operators are oriented toward the presence and arrangement of motifs. In our experiments, we begin our structure search with a single motif. We learn the parameters for this model and search HMM structure space by introducing additional motifs. We repeat this search process from different initial parameters some fixed number of times, and return the model structure and parameters that perform the best on the training set or a tuning set.

3 Results

Our task is to learn the structure and parameters of an HMM, as well as the parameters of our regression function. We hypothesize that an algorithm which uses the real-valued response associated with each input sequence to train HMM parameters is able to learn more accurate models than an approach which does not. To

[2] In practice, to make our calculations more efficient, if there is a cycle in the HMM topology that includes state c_k, we set H_k to the highest number of visits to state c_k that can reasonably be expected to occur.

test this hypothesis, we compare our path-aggregate learning approach to a slightly less sophisticated two-phase version, where we first learn the HMM parameters $\boldsymbol{\Theta}$ (using standard Baum-Welch), and then learn the parameters of our regression model $(\boldsymbol{\beta}, \sigma)$, from $\boldsymbol{\Theta}$ and the observed sequence and response data. The key difference is that the regression model is just learned once in the two-phase approach, rather than iteratively refined as described in the previous section.

Given an input sequence $\mathbf{x}$, our models predict a probability density over the response y. In order to compare our method to the baseline method, we select a single value $\hat{y}$, by calculating the Viterbi (most likely) path through the HMM and then calculating the corresponding response according to the regression model, *i.e.* $\hat{y} = \boldsymbol{\beta} \cdot \mathbf{v}$, where $\mathbf{v}$ is the counted state visits implied by the Viterbi path.

To measure the accuracy of our models, we calculate the average absolute error on held-aside test sequences: error $= \frac{1}{T} \sum_i |y_i - \hat{y}_i|$. We test our learner on both simulated data and real gene expression data from yeast. For the yeast data, we believe the gene expression measurement is a function, in part, of a combination of short DNA motifs in the gene's promoter region, to which transcription factor proteins may bind to regulate the gene's expression. For the simulated data, we create such a situation by planting known motifs in simulated DNA sequence data.

For each simulated data experiment, We generate 200-character sequences from the alphabet {a,c,g,t}. We then plant 10-character motifs in each sequence. The number of times each motif is planted comes from a Poisson distribution with $\lambda = 1$. Only two of the motifs affect the response value, which is set to $-2 + 7 \times v_1 + 3 \times v_2 + \varepsilon$, where v_i is the number of times motif i was planted in the sequence, and ε is random noise distributed normally from $N(0, 1)$. In our experiments, we vary the number of additional motifs (that do not affect response), and the "mutation rate," where a rate of r means that r characters in each motif are changed at random before the motif is planted.

The HMM model that we use is similar to the one shown in Figure 2(a), except that it varies in the number of motifs, and they are each 15 characters wide. We explore structures with one or two motifs, restarting 10 times with different initial settings. We keep the model with the highest accuracy on a held-aside tuning set. For each experiment, we generate 128 training sequences, 128 tuning sequences and 256 testing sequences, and we replicate each experiment several times.

Figure 4 shows how the accuracy of our learned models changes as a function of the mutation rate, and as

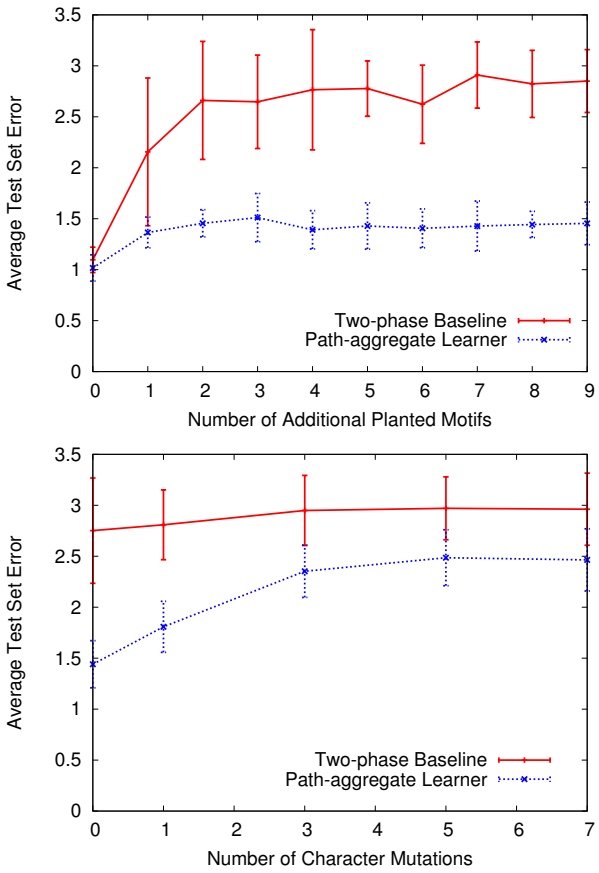


Figure 4: Test set error rate on simulated data comparing the path-aggregation learning approach to a two-phase baseline. Top: Test set error as a function of additional planted motifs that do not affect the response. Bottom: Test set error as a function of mutation rate (using five additional planted motifs).

a function of the number of additional planted motifs (apart from the two motifs that affect the response). The error rate using our approach is consistently less than that of the two-phase baseline, and tends to level off even as the number of mutations or additional motifs increases. Also, the recovery rate of the planted motifs is consistently higher using our integrated approach than that of the two-phase baseline. For instance, as we vary mutation rate and motif set size, we find that our approach returns the exact 10-character string of the motifs four times as often than the two-phase baseline. We conjecture that the reason our approach learns more accurate models than the two-phase baseline is because it is able to pick out the motifs that affect the response value instead of over-represented motifs which do not.

To determine whether our approach can aid in the discovery of motifs in real genomic data, we use data from the yeast gene expression analysis of Gasch *et al.* (2000). In these experiments, yeast cells

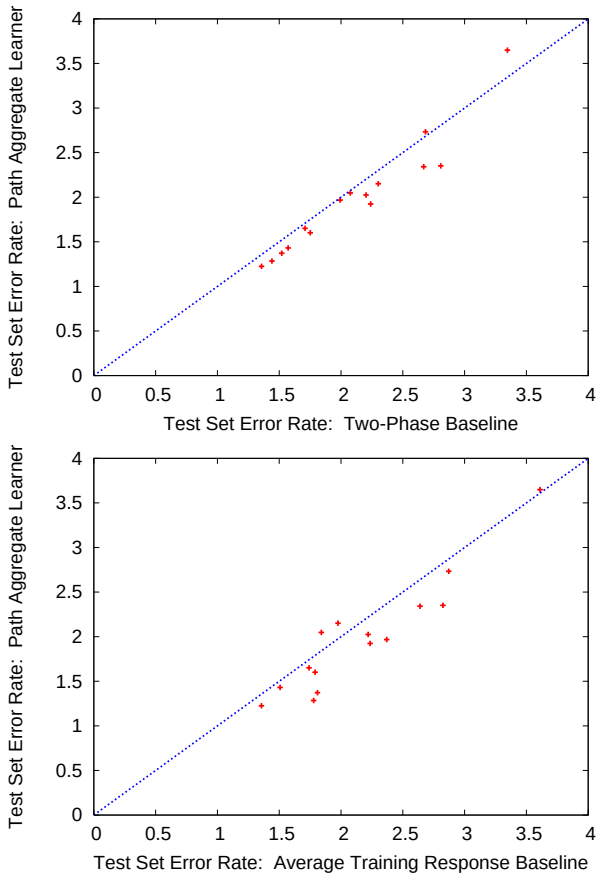


Figure 5: Test set error rate over 15 data sets, comparing our path-aggregate learner with the two-phase baseline (top) and the average training set response baseline (bottom). Each point represents one data set. Points are below the diagonal on datasets where our approach has a smaller error rate.

are put in a variety of stress conditions, such as heat shock or amino acid starvation, and measurements of gene expression are taken using microarrays to determine which genes' activity is increased or decreased specifically in these conditions. We choose 15 of these experiments that have the highest degree of differential expression and represent a variety of experimental conditions. From each of these, we select genes which are differentially expressed, and a control group with approximately the same number of genes. For each gene, we obtain 500 base pairs of promoter sequence from the University of California Santa Cruz genome browser [10].

For these data sets, we use models similar to the ones we have previously shown to be well-suited to the task of identifying motifs in promoter data [18]. An example of the HMM structure is shown in Figure 3. These models are able to represent conjunctions of motifs occuring in specific orders. Instead of counting motif occurrences, the regression model considers which *combinations* of motifs occur in each sequence. We search over the space of possible structures by incrementally adding new motifs to the existing model. Each such addition affects several parts of the HMM topology. We limit this search to a maximum of two motifs, and we find in our experiments that both our approach and the baseline method return a variety of different HMM structures. Since the initial parameter values affect the results of EM training, the motif emission parameters are initialized by sampling from the training sequences.

As one additional baseline, we include a model that always predicts the average training set expression as the predicted response: $\hat{y} = \frac{1}{D}\sum_i^D y_i$. The results are shown in Figure 5. The top panel in the figure compares our approach to the two-phase basline. The bottom panel compares against the average-expression baseline. The models learned by our path-aggregate approach are more accurate than the two-phase baseline for 13 of the 15 data sets. Eight of these 13 are statistically significant at a p-value of 0.05, using a two-tailed, paired t-test over the ten cross-validation folds. Our models are more accurate than the training set average baseline for 12 of 15 data sets (10 of these are statistically significant).

4 Conclusion

We have presented a novel approach for learning HMM models for sequence-based regression tasks. Our approach involves simultaneously learning the structure and parameters of an HMM, along with a linear regression model that maps occurrences of sequence motifs to the response variable. Our experiments indicate that integrating the processes of learning the HMM and the associated regression model yields more accurate models than a two-phase baseline regression approach which first learns the HMM and then subsequently learns a regression model. We note that this baseline is fairly sophisticated, compared to many methods for sequence-based regression, in that it does not rely on a fixed, pre-defined set of features to represent each sequence being processed.

Acknowledgements

This research was supported in part by NIH grants T15 LM007359, R01 LM07050, and R01 GM077402. The authors would like to thank Audrey Gasch, Yue Pan and Tim Durfee for help with data and analysis.

References

[1] T. Bailey and C. Elkan. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21:51–83, 1995.

[2] M. A. Beer and S. Tavazoie. Predicting gene expression from sequence. *Cell*, 117:185–198, 2004.

[3] Y. Bengio and P. Frasconi. An input output HMM architecture. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, Cambridge, MA, 1995.

[4] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268:78–94, 1997.

[5] E. M. Conlon, X. S. Liu, J. D. Lieb, and J. S. Liu. Integrating regulatory motif discovery and genome-wide expression analysis. *Proc. of the National Academy of Sciences*, 100(6):3339, 2003.

[6] A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11(12):4241–57, 2000.

[7] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[8] T. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. In *Proc. of the Seventh International Conf. on Intelligent Systems for Molecular Biology*. AAAI Press, 1999.

[9] A. Karali and I. Bratko. First order regression. *Machine Learning*, 26(2-3):147–176, 1997.

[10] D. Karolchik, A. Hinrichs, T. Furey, K. Roskin, C. Sugnet, D. Haussler, and W. Kent. The UCSC table browser data retrieval tool. *Nucleic Acids Research*, 32(1):D493–D496, 2004.

[11] S. Kramer. Structural regression trees. In *Proc. of the Thirteenth National Conf. on Artificial Intelligence*, pages 812–819. AAAI/MIT Press, 1996.

[12] A. Krogh, M. Brown, I. S. Mian, K. Sjolander, and D. Haussler. Hidden Markov models in computational biology: Applications in protein modeling. *Journal of Molecular Biology*, 238:54–61, 1994.

[13] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the Eighteenth International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, 2001.

[14] C. Lawrence, S. Altschul, M. Boguski, J. Liu, A. Neuwald, and J. Wootton. Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, 1993.

[15] C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: A string kernel for SVM protein classification. *Pacific Symposium on Biocomputing*, 7:566–575, 2002.

[16] N. Li and M. Tompa. Analysis of computational approaches for motif discovery. *Algorithms for Molecular Biology*, 1:8, 2006.

[17] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge MA, 1999.

[18] K. Noto and M. Craven. Learning probabilistic models of cis-regulatory modules that represent logical and spatial aspects. *Bioinformatics*, 23(2):e156–e162, 2007.

[19] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, 1989.

[20] E. Segal and R. Sharan. A discriminative model for identifying spatial cis-regulatory modules. In *Proc. of the Eighth Annual International Conf. on Computational Molecular Biology (RECOMB)*, pages 141–149. ACM Press, 2004.

[21] Y. Yuan, L. Guo, L. Shen, and S. Liu. Predicting gene expression from sequence: A reexamination. *PLoS Comput Biol*, 3(11):e243, Nov 2007.

[22] Q. Zhou and W. H. Wong. CisModule: *De novo* discovery of cis-regulatory modules by hierarchical mixture modeling. *Proc. of the National Academy of Sciences*, 101(33):12114–12119, 2004.

[23] C. B. Z. Zilberstein, E. Eskin, and Z. Yakhini. Using expression data to discover RNA and DNA regulatory sequence motifs. In *Regulatory Genomics: RECOMB 2004 International Workshop*. Springer-Verlag, New York, NY, 2004.

Bounding Search Space Size via (Hyper)tree Decompositions

Lars Otten and **Rina Dechter**
Bren School of Information and Computer Sciences
University of California, Irvine, CA 92697-3435, U.S.A.
{lotten,dechter}@ics.uci.edu

Abstract

This paper develops a measure for bounding the performance of AND/OR search algorithms for solving a variety of queries over graphical models. We show how drawing a connection to the recent notion of hypertree decompositions allows to exploit determinism in the problem specification and produce tighter bounds. We demonstrate on a variety of practical problem instances that we are often able to improve upon existing bounds by several orders of magnitude.

1 INTRODUCTION

This paper develops a measure for bounding the performance of search algorithms for solving a variety of queries over graphical models. It has been known for a while that the complexity of inference algorithms (e.g., join-tree clustering, variable elimination) is exponentially bounded by the tree width of the graphical model's underlying graph. The base of the exponent is often taken to be the maximum domain size.

More accurate bounds were derived by looking at the respective domain sizes and their product in each cluster in of tree decomposition of the underlying graph [Kjærulff, 1990]. These tighter bounds were used in selecting good variable orderings, for example. It was recently shown that these bounds are also applicable to search algorithms that explore the context-minimal AND/OR search graph [Dechter and Mateescu, 2007].

The shortcoming of these bounds is that they are completely blind to context-sensitivity hidden in the functions of the graphical model and especially determinism. When a problem possesses high levels of determinism, its tree width bound can be large while its search space can be extremely pruned, due to propagation of inconsistencies across functions.

Part of this shortcoming in worst-case complexity bounds is addressed by the more recent concept of hypertree decompositions [Gottlob et al., 2000]. It was shown that the maximum number of functions in the clusters of a hypertree decomposition (the hypertree width) exponentially bounds the problem complexity for constraint inference, a result that was extended to general graphical model inference in [Kask et al., 2005]. The base of the exponent in this case is the relation tightness, thus allowing the notion of determinism to play a role. However, in practice this bound often turns out to be far worse than the tree width bound, unless the problem exhibits substantial determinism [Dechter et al., 2008].

The contribution of this paper is in combining both ideas to tighten the existing bounds, using the relationship between AND/OR graph search and tree decompositions. Starting with the tree width bound, we show that one can also incorporate the concept of hypertree decompositions by greedily covering variables with tight functions. This yields better bounds on the number of nodes in the search graph, which translates directly to search complexity.

Tighter bounds are desirable for a number of reasons:

1. We can better predict parameters of the algorithm ahead of time (primarily the variable ordering for search), fitting the algorithm to the problem.
2. It enables us to dynamically update parameters during search, e.g., for dynamic variable orderings.
3. In a distributed setup, search can often be implemented as centralized conditioning followed by independent solving of the conditioned subproblems on different machines. Better bounds can help in balancing these two phases by varying the size of the central conditioning set [Silberstein et al., 2006].

We provide extensive empirical results on 112 probabilistic problem instances and 30 weighted constraint satisfaction problems. We show that exploiting determinism has a significant effect for a number of problem classes. We furthermore compare our bound to the exact size of the search space on a subset of feasible instances, and show that it can be very tight in some cases.

An approach that is related but orthogonal to the work here is described in [Zabiyaka and Darwiche, 2006], where the standard complexity measure of tree width is refined by taking into account functional dependencies – i.e., knowing one set of variables determines the values of another set. [Fishelson et al., 2005] develops a bound specifically for an interleaved variable elimination and conditioning algorithm on linkage analysis problems.

Section 2 provides the background and definitions. Section 3 discusses hypertree decompositions and related complexity bounds. In Section 4 we introduce our new bounding scheme, for which Section 5 provides empirical evaluation. Section 6 concludes.

2 PRELIMINARIES AND DEFINITIONS

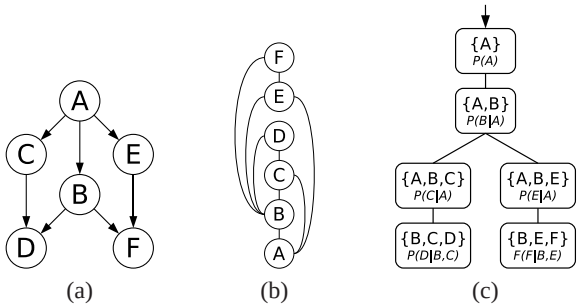
In the following we will assume a *graphical model* given as a set of variables $X = \{x_1, \ldots, x_n\}$, their finite domains $D = \{D_1, \ldots, D_n\}$, a set of functions $F = \{f_1, \ldots, f_m\}$, each of which is defined over a subset of X, and a combination operator (typically sum, product or join) over all functions. Together with an marginalization operator such as $\min_X$ and $\max_X$ we obtain a *reasoning problem*.

The special cases of reasoning tasks which we have in mind are belief networks, (weighted) constraint networks or mixed networks that combine both. The primary tasks over belief networks are belief updating and finding the most probable explanation. They are often specified using conditional probability functions defined on each variable and its parents in a given directed acyclic graph (see Figure 1(a)), and use multiplication and summation or maximization as the combination and marginalization operators [Kask et al., 2005]. For constraint networks we are mainly concerned with problems of finding or enumerating solutions; they are defined using relations as functions, and relational join and projection as the combination and marginalization operators, respectively. For weighted constraint networks one typically has real-valued functions and summation and minimization as combination and marginalization operators, respectively.

2.1 EXPRESSING STRUCTURE

If one wants to analyze the complexity of a given problem instance, it has proven useful to look at the underlying structure of interactions between variables:

DEFINITION **2.1** *The* **hypergraph** *of a graphical model is a pair $H = (V, S)$, where the vertices are the problem variables ($V = X$) and where $S = \{S_1, ..., S_r\}$ is a set of subsets of V, called hyperedges, which represent the scopes of the functions in the problem ($S_i = scope(f_i)$). The* **primal graph** *of a hypergraph $H = (V, S)$ is an undirected graph $G = (V, E)$ such that there is an edge $(u, v) \in E$ for any two vertices $u, v \in V$ that appear in the same hyperedge (namely, there exists S_i, s.t., $u, v \in S_i$). The* **dual graph** *of a hypergraph $H = (V, S)$ is an undirected graph $G = (S, E)$ that has a vertex for each hyperedge, and there is an edge $(S_i, S_j) \in E$ when the corresponding hyperedges share a vertex ($S_i \cap S_j \neq \emptyset$).*



(a) (b) (c)

Figure 1: Example belief network, its triangulated primal graph along ordering $d = A, B, C, D, E, F$, and the corresponding bucket tree decomposition.

DEFINITION **2.2** *A hypergraph is a* **hypertree**, *also called* **acyclic hypergraph**, *if and only if its dual graph has an edge subgraph that is a tree, such that all the nodes in the dual graph that contain a common variable form a connected subgraph.*

It is well-known that problems whose underlying graph have tree structure can be solved efficiently [Pearl, 1988]. If this is not the case, we aim to transform the problem into an equivalent one that exhibits tree structure [Lauritzen and Spiegelhalter, 1988, Dechter and Pearl, 1989, Kask et al., 2005]. Intuitively, we do this by grouping variables and the functions over them into clusters that can be arranged as a tree:

DEFINITION **2.3** *Let X, D, F be the variables, domains and functions of a reasoning problem $\mathcal{P}$. A* **tree decomposition** *of $\mathcal{P}$ is a triple $\langle T, \chi, \psi \rangle$, where $T = (V, E)$ is a tree and χ and ψ are labeling functions that associate with each vertex $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq F$, that satisfy the following conditions:*

1. *For each $f_i \in F$, there is at least one vertex $v \in V$ such that $f_i \in \psi(v)$.*
2. *If $f_i \in \psi(v)$, then $scope(f_i) \subseteq \chi(v)$.*
3. *For each variable $X_i \in X$, the set $\{v \in V | X_i \in \chi(v)\}$ induces a connected subtree of T. This is also called the running intersection property.*

The **tree width** *of a tree decomposition $\langle T, \chi, \psi \rangle$ is $w = \max_v |\chi(v)| - 1$. The tree width w^* of $\mathcal{P}$ is the minimum tree width over all its tree decompositions.*

The problem of finding the tree decomposition of minimal tree width is known to be NP-complete. To obtain tree decompositions in practice, one can apply a triangulation algorithm to the problem's primal graph along an ordering

and then construct the *bucket tree* by extracting and connecting the cliques for each variable, as described for instance in [Pearl, 1988]. The ordering to use as the basis for the triangulation algorithm is often computed heuristically.

Example 2.1 *Assume the belief network in Figure 1(a) over variables $X = \{A, B, C, D, E, F\}$ is given. We pick the ordering $d = A, B, C, D, E, F$ and triangulate the primal graph as shown in Figure 1(b). If we extract each variable's* bucket *– the variable and its earlier neighbors – we obtain the tree decomposition shown in Figure 1(c), the* bucket tree decomposition*.*

2.2 SOLVING REASONING PROBLEMS

Two principal methods exist to solve reasoning problems, search (e.g., depth-first branch-and-bound, best-first search) and inference (e.g., variable elimination, join-tree clustering). Both can be shown to be time and space exponential in the problem instance's tree width [Lauritzen and Spiegelhalter, 1988, Dechter and Pearl, 1989, Kask et al., 2005, Dechter and Mateescu, 2007], with a dominant factor of k^w, where k denotes the maximum domain of the problem variables.

2.2.1 Search

Search-based algorithms traverse the problem *search space*. Given a variable ordering d, the simplest way to perform search is to instantiate variables one at a time. This will define a search tree, where each node represents a state in the space of partial assignments. Leaf nodes signify either full solutions or dead ends. Standard depth-first algorithms typically have time complexity exponential in the number of variables and require linear space. If memory is available, one can apply caching to traversed nodes and retrieve their values when "similar" nodes are encountered.

These elementary search spaces, however, don't fully capture the structure of the underlying graphical model. Introducing *AND* nodes into the search space can exploit independence of subproblems by effectively conditioning on values, thus avoiding redundant computation [Dechter and Mateescu, 2007]. Since the size of the *AND/OR search tree* may be exponentially smaller than the traditional OR search one, any algorithm exploring the AND/OR space enjoys a better computational bound.

Example 2.2 *Figure 2 depicts the AND/OR search tree for the problem introduced in Example 2.1 if we assume binary variable domains and the ordering $d = A, B, C, D, E, F$. The AND nodes for variable B each have two OR children, expressing that at this point the problem decomposes into independent subproblems, rooted at C and E, respectively.*

We can equally apply caching techniques to an algorithm exploring the AND/OR search tree. As a result this algorithm will effectively explore the *AND/OR search graph*. With caching, identical subproblems are recognized based on their context, which is a graphical model parameter that denotes the part of the search tree above that is relevant to the subproblem below.

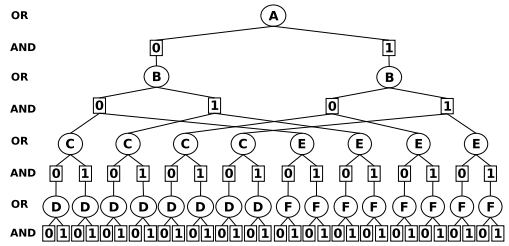


Figure 2: The AND/OR search tree for the example problem, assuming binary variables.

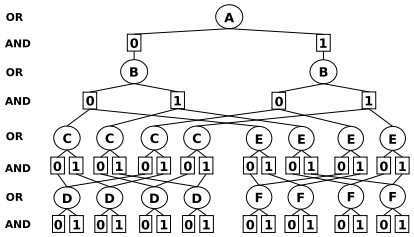


Figure 3: The AND/OR search graph explored by AND/OR search with caching.

Caching will avoid redundant computations, thus reducing time complexity, at the cost of increased memory requirements. By varying the maximum size of contexts to cache on, this tradeoff can be fine-tuned. Assuming full caching, search has been shown to exhibit both time and space complexity exponential in the problem's tree width.

Example 2.3 *If we extend AND/OR search with caching, given the problem in Example 2.1 it will explore the AND/OR search graph shown in Figure 3. Note how the child nodes of variables C and E are merged. This expresses the fact that in the example the subproblems rooted at D and F, as children of C and E, respectively, are independent of the value of A further up.*

2.3 EXPLOITING DETERMINISM

In practice, however, problem instances across many domains will exhibit a significant degree of determinism (e.g., disallowed tuples in constraint problems, zero probability entries in belief networks). Search algorithms detect the resulting inconsistencies early in the search process and prune the respective portion of the search space. This can lead to significant savings in running time, but is not reflected in the standard worst-case bounds described above.

To exploit determinism in the context of variable elimination, the concept of (generalized) hypertree decompositions has been introduced for constraint networks in

[Gottlob et al., 2000]. As a subclass of tree decompositions, it was shown that it provides a stronger indicator of tractability than the tree width.

DEFINITION **2.4** *Let $\mathcal{T} = \langle T, \chi, \psi \rangle$, where $T = (V, E)$ be a tree decomposition of a reasoning problem $\mathcal{P}$ over a graphical model with variables X, their domains D and functions F. $\mathcal{T}$ is a* **hypertree decomposition** *of $\mathcal{P}$ if the following additional condition is satisfied:*

4. *For each $v \in V$, $\chi(v) \subseteq \bigcup_{f_j \in \psi(v)} scope(f_j)$.*

The **hypertree width** *of a hypertree decomposition is $hw = \max_v |\psi(v)|$. The hypertree width hw^* of $\mathcal{P}$ is the minimum hypertree width over all its hypertree decompositions.*

To analyze the complexity of algorithms operating on hypertree decompositions, we introduce the notion of *tightness* of a function or relation:

DEFINITION **2.5** *The* **tightness** *t of a function f is the number of relevant tuples (e.g., allowed tuples in constraints, nonzero entries in conditional probability tables).*

The motivation behind this is to store and process the function in a "compressed" form, with only the t relevant tuples. Given a hypertree decomposition, one can then modify an inference algorithm to make use of these compact representations when computing the messages to be passed.

In [Gottlob et al., 2000] the complexity of processing a hypertree decomposition for solving a constraint satisfaction problem is shown to be exponential in hw^*, with a dominant factor of t^{hw^*}. This result was extended in [Kask et al., 2005] to any graphical model that is absorbing relative to 0. (A graphical model is absorbing relative to a 0 element if its combination operator has the property that $x \bigotimes 0 = 0 \ \forall x$; for example, multiplication has this property while summation does not.)

3 HYPERTREE WIDTH BOUNDS FOR INFERENCE

In this section we briefly explore whether the bounds based on hypertree width can provide a practical improvement over the established tree width bounds described above. To that end we recently looked at a selection of over 140 problem instances from various domains [Dechter et al., 2008].

We used the code developed for [Dermaku et al., 2005], which is available online. It generates a tree decomposition along a minfill ordering and extends it to a (generalized) hypertree decomposition by applying a greedy heuristic.

We used the lowest tree width w and hypertree width hw out of 20 runs as a basis for our investigation: For every problem instance, we compute the dominant factors of the two worst-case complexity bounds, i.e., k^w and t^{hw}, where k denotes the maximum domain size and t the maximum function tightness in the problem instance. In order for the hypertree decomposition to provide a better bound than the tree decomposition, t^{hw} should be significantly smaller than k^w.

Intuitively, it is clear that $t \in \mathcal{O}(k^r)$, where r is the maximum function arity of the problem. Hence we should expect that only when the function table contains many irrelevant values (e.g., zeros in probability tables), the hypertree width bound can be superior.

And indeed, out of all the instances we evaluated, only for five of them was t^{hw} less than k^w, whereas it was orders of magnitude worse on almost all of the remaining problems. Looking at the instances in more detail, it becomes evident that in most of them the functions are not sufficiently tight and often have highly intersecting scopes, which renders the hypertree width bound ineffective.

4 SEARCH SPACE ESTIMATION

Even though the results in [Dechter et al., 2008] suggest that complexity bounds based on hypertree width are often not competitive in practice, the idea of exploiting determinism remains promising. Furthermore, while all considerations in section 3 were targeted at inference algorithms, we are in particular interested in estimating how determinism in a problem will impact the size of the search space discussed in Section 2.2.1, since search is a widespread method in practice. To that end, we will aim to upper bound the size of the AND/OR context minimal search graph, which is explored by AND/OR search augmented with caching, as described in Section 2.

4.1 TREE DECOMPOSITION CORRESPONDENCE

Assume that a variable ordering $d = x_1, \ldots, x_n$ is fixed and that search instantiates variables first to last (while inference would proceed last to first). We note that the way the search space is decomposed by AND/OR search with caching can be represented by the bucket tree decomposition along the same ordering. For details we refer to [Mateescu and Dechter, 2005], to illustrate we revisit our previous example:

Example 4.1 *Consider the bucket tree decomposition in Figure 1(c) and the AND/OR search graph in Figure 3. It is easy to see that the decomposition clusters can be related to the "layers" of the search graph, i.e., the nodes associated with a variable and its values, as shown in Figure 4. For instance, the cluster $\{B, C, D\}$ represents the search layer for variable D and the fact that the subproblem only depends on B and C – and not A.*

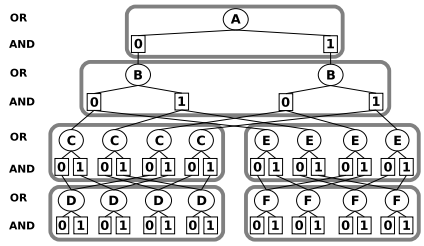


Figure 4: The AND/OR search graph with clusters corresponding to the bucket tree decomposition in Figure 1(c).

Based on this observation, we can also partition the search space into "clusters", according to the corresponding bucket tree decomposition. Our approach of upper bounding the size of the entire search space will then be to bound the portion of the search space in each cluster, subsequently summing over the clusters (for simplicity we consider only the AND nodes of the search space, since OR nodes are actually not implemented as such in practice). We note that a cluster in a bucket tree decomposition will always correspond to exactly one variable's layer in the AND/OR search graph (due to the way it is constructed).

4.2 BOUNDING CLUSTER SIZE

A straightforward upper bound is obtained by multiplying the domain sizes of all variables in the cluster [Kjærulff, 1990]. If the set of clusters is given by $C = \{C_1, \ldots, C_n\}$, $C_k \subseteq X$, summing over clusters gives

$$twb := \sum_{k=1}^{n} \prod_{x_i \in C_k} |D_i| \,.$$

Note that this is very closely related to the worst case complexity, since the tree width is just the maximum number of variables in any cluster of the decomposition. This, however, does not take determinism into account.

If the bucket tree decomposition we are working with is also a hypertree decomposition (meaning all variables in a cluster are covered by the functions in that cluster), we can take the product $\prod_i t_i$ as an upper bound to the number of nodes in that cluster, where t_i is the tightness of the i-th function in it. (This is closely related to the complexity bound on hypertree decompositions as outlined above.)

Taking this one step further, if the bucket tree decomposition at hand does not satisfy the additional hypertree decomposition condition, we can compute the product over the tightness of each function in the cluster and multiply this by the domain sizes of the uncovered variables, to account for the lack of information about these variables.

However, since the scopes of the different functions in a cluster can overlap, this tightness based bound will typically be worse than the simple "product of domain sizes" (as we already observed). Consequentially, we use the latter as a starting point and exploit tight functions to improve upon it, therefore combining the concept of tree and hypertree decompositions.

Algorithm *GreedyCovering*
Input: Cluster C_k containing variables X_k and functions F_k, with $x_i \in X_k$ having domain size d_i and $f_j \in F_k$ having tightness t_j
Output: A subset of F_k (forming a partial covering of X_k)
Init: *Uncov* $= X_k$, *Covering* $= \emptyset$
(1) Find j^* that minimizes $r_j = t_j / \prod_{x_k \in I_j} d_k$, where $I_j =$ *Uncov* $\cap\, scope(f_j)$.
(2) If $r_{j^*} \geq 1$, terminate with current covering.
(3) Add f_{j^*} to *Covering* and set *Uncov* $:=$ *Uncov* $\backslash\, scope(f_{j^*})$.
(4) If *Uncov* $= \emptyset$, terminate with current covering.
(5) Goto (1).

Figure 5: Greedy covering algorithm for a single cluster.

In essence this can be seen as a weighted variant of the well-known SET COVER problem, where one aims to cover a set or vertices by as few as possible subsets from a set of given subsets of the variables. The problem is generally NP-complete, but simple greedy approximations exist [Johnson, 1973], which give rise to our method:

Start with an empty covering (if we assume dummy unary functions over uncovered variables, this is equivalent to the bound twb). Then, for each function f_j in the cluster, compute the *coverage ratio* r_j as follows: Divide the function's tightness t_j by the product over the domain sizes of the variables that have not yet been covered and are in the scope of f_j. Pick the function for which the coverage ratio is the lowest and add it to the covering. Repeat this for as long as we can still find a function with a coverage ratio less than 1. The algorithm is given in Figure 5.

It will produce a set of functions as the covering, but might leave some variables uncovered. As before, we can multiply the tightness of the functions in the covering and the domain sizes of the uncovered variables to obtain an upper bound on the number of nodes in the cluster.

It is worth noting that we are not limited to functions from the decomposition cluster in question, but we can include any function from the clusters higher up in the rooted tree decomposition (since their scope will have been fully instantiated at this point of the search).

Proposition 1 *Executing algorithm* GreedyCovering *for each cluster of the bucket tree decomposition and summing up, we obtain an upper bound on the number of nodes in the AND/OR search graph, which we denote:*

$$hwb := \sum_{k=1}^{n} \left(\prod_{f_j \in Cov(C_k)} t_j \;\cdot \prod_{x_i \in\, C_k \backslash Cov(C_k)} |D_i| \right) ,$$

where $Cov(C_k)$ is the set of functions returned by the algorithm GreedyCovering *for cluster C_k.*

THEOREM **4.2** *Given a reasoning problem with n variables and m relations with maximal tightness t. If the bucket tree decomposition along a given variable ordering d has tree width w, the complexity of computing the bound hwb is $\mathcal{O}(n \cdot w \cdot m)$ time-wise and $\mathcal{O}(m)$ space-wise.*

Proof. The time complexity of algorithm *GreedyCovering* is linear both on the number of variables in the cluster as well as in the number of functions considered, i.e., worst-case $\mathcal{O}(w \cdot m)$. Keeping track of the coverage ratio of each function requires $\mathcal{O}(m)$ space. Iterating and summing over all n clusters results in the stated asymptotic bounds. □

Example 4.3 *Assume we have a cluster containing 3 variables X, Y, Z with domain sizes $d_X = 4$, $d_Y = 4$, and $d_Z = 3$, as well as 2 functions $f_1(X, Y)$ and $f_2(Y, Z)$ with tightness $t_1 = 9$ and $t_2 = 11$. The twb bound on the number of nodes in this cluster is $d_X \cdot d_Y \cdot d_Z = 48$. In the first iteration of the greedy covering algorithm for hwb the gain ratios of f_1 and f_2 will be computed as $\frac{9}{16}$ and $\frac{11}{12}$, respectively. Therefore f_1 will be added to the covering, leaving only Z uncovered. The next gain ratio of f_2 will be computed as $\frac{11}{3}$, which is greater than 1. Therefore the algorithm terminates and f_2 will not be part of the covering. The hwb bound for this cluster will then be $t_1 \cdot d_Z = 9 \cdot 3 = 27$.*

5 EXPERIMENTAL RESULTS

For empirical evaluation we return to the problem instances that were described in [Dechter et al., 2008] (see also Section 3). These comprise 112 belief networks from areas such as coding networks, dynamic Bayesian networks, genetic linkage instances and CPCS medical diagnosis networks. We also evaluated 30 weighted constraint network instances. All problem instances are available online[1].

For every instance, we report the number of variables n, the maximum variable domain size k, the maximum function arity r, and the average tightness ratio tr (defined as the average percentage of relevant tuples in a full function table).

On each problem instance we run our bounding method along 100 different minfill orderings (with random tie breaking) and record the lowest bound, with w as the tree width of the bucket tree decomposition. For every instance we then compute the asymptotic worst-case bound for the search space size, which is $n \cdot k^{w+1}$ (cf. [Dechter and Mateescu, 2007], adapted for AND nodes).

Consulting the bucket tree decomposition for a more fine-grained analysis (still without considering determinism) gives the bound twb. We then apply our covering heuristic to exploit determinism and obtain the bound hwb. (Note that these three bounds can produce very large integers, therefore we report them as their $\log_{10}$ in Table 1.) Even for the larger problem instances this bound computation takes only a few hundred milliseconds on a 2.66 GHz CPU.

[1] http://graphmod.ics.uci.edu/

5.1 BOUND IMPROVEMENTS

If we compare the values for $n \cdot k^{w+1}$ and twb, we can see that the bound improves for every problem instance by at least one order of magnitude, but often more than that (recall that the table shows $\log_{10}$ of the bounds). For most pedigree genetic linkage instances, for example, the reduction is ten orders of magnitude or more. Similar results hold for the dynamic Bayesian networks we tested on, with many orders of magnitude improvement.

It seems that problems with a higher number of variables benefit the most from the fine-grained analysis. This makes sense if one considers the fact that the worst-case bound will greatly overestimate the size of almost all clusters, since in practice the tree decomposition contains only very few clusters of full tree width.

If we try to exploit determinism by going from the twb to hwb bound, there is, just looking at the problem parameters, no obvious indicator for when the bound will improve: On pedigree instances, for example, the decrease is not very significant, although these problems exihibit some determinism. On digital circuits, on the other hand, with an average of 50% determinism, the bound improves another 3 to 4 orders of magnitude over twb.

On almost all weighted CSP instances we were able to lower the bound by exploiting determinism, often by orders of magnitude. For example, on the satellite scheduling problem 408b the twb bound of $83,206,198,094$ decreases to a hwb value of $248,197$. A significantly tighter bound is also achieved on randomly generated k-trees where the probability tables were forced to exihibit determinism, with for example $twb = 29,983,742$ decreasing to $hwb = 1,528$ on problem BN_107.

The crucial point here seems to be at which point during the search functions with high determinism will have their scope fully instantiated, i.e., at which point they become available to our covering heuristic. This is not predictable by only looking at the instance parameters but will require a more detailled look at the guiding bucket tree decomposition instead.

5.2 BOUND EVALUATION

Most problem instances are too big to compute the exact size of the context minimal AND/OR search space, which would be equivalent to solving the problem (for solution counting or computing $P(e)$). But for some of the smaller instances this is actually feasible, which gives us the option of testing how tight our bound is.

						$\log_{10}$		
instance	n	k	r	tr	w	nk^{w+1}	twb	hwb
Grid networks								
90-10-1	100	2	3	0.58	9	5.01	4.06	3.92
90-14-1	196	2	3	0.55	15	7.11	5.63	5.58
90-16-1	256	2	3	0.56	17	7.83	6.25	6.15
90-24-1	576	2	3	0.55	15	7.58	5.91	5.76
90-24-1e20	576	2	3	0.55	31	12.39	10.12	10.12
90-26-1e40	676	2	3	0.55	29	11.86	9.88	9.83
90-30-1e60	900	2	3	0.55	37	14.39	11.96	11.95
90-34-1e80	1156	2	3	0.56	39	15.10	12.66	12.53
90-38-1e120	1444	2	3	0.55	43	16.40	13.97	13.69
Dynamic Bayesian Networks								
BN_21	2843	91	4	0.49	6	17.17	8.50	7.82
BN_23	2425	91	4	0.47	4	13.18	7.37	6.59
BN_25	1819	91	4	0.53	4	13.06	7.17	6.77
BN_27	3025	5	7	1.00	9	10.47	7.33	7.33
BN_29	24	10	6	1.00	5	7.38	6.16	6.16
Grid networks								
BN_31	1156	2	3	0.56	35	13.90	11.53	11.47
BN_33	1444	2	3	0.56	37	14.60	12.42	12.26
BN_35	1444	2	3	0.55	38	14.90	12.51	12.25
BN_37	1444	2	3	0.55	40	15.50	13.01	12.99
BN_39	1444	2	3	0.56	38	14.90	12.63	12.57
BN_41	1444	2	3	0.56	40	15.50	13.09	12.99
Digital circuits								
BN_48	661	2	5	0.51	43	16.07	13.79	10.44
BN_50	661	2	5	0.51	43	16.07	13.79	10.69
BN_52	661	2	5	0.51	41	15.46	13.39	9.86
BN_54	561	2	5	0.53	48	17.50	15.43	13.05
BN_56	561	2	5	0.53	51	18.40	16.19	14.04
BN_58	561	2	5	0.53	50	18.10	15.72	13.09
BN_60	540	2	5	0.53	55	19.59	17.35	14.43
BN_62	667	2	5	0.51	42	15.77	13.93	10.73
BN_64	540	2	5	0.53	50	18.08	15.86	14.07
BN_66	440	2	5	0.55	59	20.71	18.82	16.15
BN_68	440	2	5	0.55	57	20.10	18.08	15.48
CPCS medical diagnosis								
BN_79	54	2	10	1.00	10	5.04	4.23	4.23
BN_81	360	2	12	0.93	16	7.67	5.75	5.74
BN_83	360	2	12	0.97	18	8.28	6.31	6.31
BN_85	360	2	12	0.99	19	8.58	6.61	6.61
BN_87	422	2	18	0.98	21	9.25	7.36	7.36
BN_89	422	2	18	0.97	17	8.04	6.33	6.33
BN_91	422	2	18	0.98	21	9.25	7.31	7.31
BN_93	422	2	18	0.97	20	8.95	6.99	6.99
Randomly generated belief networks								
BN_95	53	3	4	1.00	15	9.36	7.01	7.01
BN_97	54	3	4	1.00	15	9.37	7.13	7.13
BN_99	57	3	4	1.00	16	9.87	7.70	7.70
BN_101	58	3	4	1.00	15	9.40	7.00	7.00
BN_103	76	3	4	1.00	17	10.47	7.43	7.43
Randomly generate partial k-trees with forced determinism								
BN_105	50	2	21	0.60	18	7.42	6.39	2.54
BN_107	50	2	21	0.59	21	8.32	7.48	3.18
BN_109	50	2	20	0.62	20	8.02	7.12	3.49
BN_111	50	2	20	0.63	19	7.72	6.92	3.31
BN_113	50	2	21	0.62	21	8.32	7.28	3.38
Randomly generate partial k-trees without forced determinism								
BN_115	50	2	19	1.00	20	8.02	7.07	7.07
BN_117	50	2	20	1.00	18	7.42	6.43	6.43
BN_119	50	2	19	1.00	19	7.72	6.60	6.60
BN_121	50	2	19	1.00	19	7.72	6.75	6.75
BN_123	50	2	20	1.00	18	7.42	6.46	6.46
BN_125	50	2	18	1.00	19	7.72	6.70	6.70
Digital circuits (WCSP)								
c432	432	2	10	0.61	27	11.06	8.90	8.90
c499	499	2	6	0.63	23	9.92	8.03	7.25
c880	880	2	5	0.64	24	10.47	7.93	6.90
s1196	561	2	5	0.82	51	18.40	16.37	13.91
s1238	540	2	5	0.87	54	19.29	17.04	14.82
s1423	748	2	5	0.78	22	9.80	7.54	7.54
s1488	667	2	5	0.90	44	16.37	14.34	10.99
s1494	661	2	5	0.91	44	16.37	14.34	10.76
s386	172	2	5	0.82	18	7.96	6.46	4.99
s953	440	2	5	0.80	62	21.61	19.69	16.30
Mastermind puzzle game (WCSP)								
03_08_03	1220	2	3	0.85	20	9.41	7.28	5.29
03_08_04	2288	2	3	0.87	30	12.69	10.37	7.96
03_08_05	3692	2	3	0.88	38	15.31	12.73	8.77
04_08_03	1418	2	3	0.85	24	10.68	8.57	6.68
04_08_04	2616	2	3	0.88	36	14.56	11.99	9.15
10_08_03	2606	2	3	0.88	48	18.17	15.80	13.60

						$\log_{10}$		
instance	n	k	r	tr	w	nk^{w+1}	twb	hwb
Coding networks								
BN_126	512	2	5	0.87	53	18.96	16.81	16.81
BN_127	512	2	5	0.88	57	20.17	17.86	17.86
BN_128	512	2	5	0.88	48	17.46	15.48	15.48
BN_129	512	2	5	0.88	52	18.66	16.66	16.66
BN_130	512	2	5	0.88	54	19.27	16.98	16.98
BN_131	512	2	5	0.87	48	17.46	15.42	15.42
BN_132	512	2	5	0.88	49	17.76	16.00	16.00
BN_133	512	2	5	0.87	54	19.27	17.26	17.26
BN_134	512	2	5	0.87	52	18.66	16.41	16.41
CPCS medical diagnosis								
cpcs54	54	2	10	1.00	12	5.65	4.68	4.68
cpcs179	179	4	9	1.00	7	7.07	5.04	5.04
cpcs360b	360	2	12	1.00	16	7.67	5.50	5.50
cpcs422b	422	2	18	0.99	22	9.55	7.51	7.51
Genetic linkage								
pedigree1	334	4	5	0.79	15	12.06	6.88	6.85
pedigree18	1184	5	5	0.81	20	17.75	7.58	7.58
pedigree20	437	5	4	0.79	22	18.72	9.42	9.20
pedigree23	402	5	4	0.80	27	22.18	11.73	10.85
pedigree25	1289	5	5	0.83	24	20.58	9.18	9.18
pedigree30	1289	5	5	0.82	21	18.49	7.95	7.95
pedigree33	798	4	5	0.81	30	21.57	11.37	10.20
pedigree37	1032	5	4	0.82	21	18.39	10.84	10.74
pedigree38	724	5	4	0.78	16	14.74	10.72	10.52
pedigree39	1272	5	4	0.85	20	17.78	8.21	8.12
pedigree42	448	5	4	0.79	23	19.43	10.66	10.14
pedigree50	514	6	4	0.77	18	17.50	11.57	11.53
pedigree7	1068	4	4	0.83	32	22.90	11.94	11.71
pedigree9	1118	7	4	0.79	26	25.87	9.88	9.86
pedigree13	1077	3	4	0.83	34	19.73	12.04	11.99
pedigree19	793	5	5	0.78	23	19.67	10.00	9.99
pedigree31	1183	5	5	0.81	30	24.74	11.77	11.77
pedigree34	1160	5	4	0.83	32	26.13	12.16	12.16
pedigree40	1030	7	5	0.80	29	28.37	12.38	12.38
pedigree41	1062	5	5	0.80	32	26.09	12.26	12.05
pedigree44	811	4	5	0.80	26	19.16	10.13	9.98
pedigree51	1152	5	4	0.82	38	30.32	12.89	12.84
Digital circuits								
c432.isc	432	2	10	0.54	20	8.96	6.96	5.43
c499.isc	499	2	6	0.54	19	8.72	6.98	5.19
s386.scan	172	2	5	0.54	16	7.35	5.71	4.61
s953.scan	440	2	5	0.54	26	10.77	8.85	6.50
Various networks								
Barley	48	67	5	0.98	7	16.29	7.26	7.26
Diabetes	413	21	3	0.45	4	9.23	7.10	6.83
hailfinder	56	11	5	0.84	4	6.96	4.01	3.78
insurance	27	5	4	0.84	6	6.32	4.50	4.45
Mildew	35	100	4	0.62	4	11.54	6.57	6.03
Munin1	189	21	4	0.49	11	18.14	8.31	8.15
Munin2	1003	21	4	0.48	8	14.90	6.86	6.70
Munin3	1044	21	4	0.47	9	16.24	6.95	6.72
Munin4	1041	21	4	0.47	9	16.24	7.54	7.27
Pigs	441	3	3	0.70	10	7.89	5.90	5.89
Water	32	4	6	0.58	10	8.13	6.66	6.20
Genetic linkage								
fileEA0	381	4	4	0.81	7	7.40	3.92	3.68
fileEA1	836	5	4	0.82	11	11.31	4.68	4.16
fileEA2	979	5	4	0.82	11	11.38	4.86	4.48
fileEA3	1122	5	4	0.82	13	12.84	5.19	4.70
fileEA4	1231	5	4	0.82	13	12.88	5.15	4.71
fileEA5	1515	5	4	0.82	12	12.27	5.27	4.94
fileEA6	1816	5	4	0.82	14	13.74	5.85	5.36
Satellite scheduling (WCSP)								
29	82	4	2	0.74	14	10.94	8.41	6.01
42b	190	4	2	0.78	18	13.72	10.46	7.19
54	67	4	3	0.75	11	9.05	6.33	4.49
404	100	4	3	0.74	19	14.04	7.65	3.83
408b	200	4	2	0.75	24	17.35	10.58	5.40
503	143	4	3	0.76	9	8.18	5.77	4.22
505b	240	4	2	0.75	16	12.62	8.87	6.67
Radio frequency assignment (WCSP)								
C6-sub0	16	44	2	0.31	7	14.35	12.99	10.20
C6-sub1-24	14	24	2	0.26	9	14.95	14.12	9.67
C6-sub1	14	44	2	0.24	9	17.58	16.75	11.76
C6-sub2	16	44	2	0.24	10	19.28	18.01	12.05
C6-sub3	18	44	2	0.26	10	19.33	18.02	12.48
C6-sub4-20	22	20	2	0.34	11	16.95	15.94	11.50
C6-sub4	22	44	2	0.30	11	21.06	19.78	15.30

Table 1: Results for experiments on 112 Bayesian networks and 30 weighted CSP instances.

instance	nk^{w+1}	twb	hwb	$\#cm$
90-10-1	204,800	14,154	11,908	11,519
90-14-1	25,690,112	804,822	689,786	683,823
90-16-1	134,217,728	2,637,878	2,335,466	188,625
90-24-1	150,994,944	1,286,726	1,115,509	52,802
cpcs54	442,368	48,842	48,842	48,842
cpcs179	11,730,944	110,560	110,512	110,512
cpcs360b	47,185,920	319,724	319,623	319,623
c432.isc	1,811,939,328	10,793,946	685,001	683,823
c499.isc	1,046,478,848	12,089,118	189,637	188,625
s386.scan	45,088,768	802,526	94,830	52,802
s953.scan	472,446,402,560	2,685,782,044	4,547,508	236,430
fileEA0	24,969,216	9,454	5,316	4,774
fileEA1	1,020,507,812,500	63,520	18,444	14,057
fileEA2	5,975,341,796,875	167,630	33,851	24,203
fileEA3	34,240,722,656,250	253,170	58,147	45,052
fileEA4	939,178,466,796,875	675,230	53,214	30,868
fileEA5	9,246,826,171,875	282,454	101,825	36,146
fileEA6	6,927,490,234,375,000	2,460,002	333,198	62,041

Table 2: Comparison of bounds to exact search space size $\#cm$.

Table 2 shows the upper bounds nk^{w+1}, twb, and hwb (this time not in their $\log_{10}$), as well as the exact number of AND nodes in the actual context-minimal search graph. The values in each row were obtained on the same minfill ordering (not neccessarily the one used for Table 1).

For smaller instances the bound we compute turns out to be rather tight (note that CPCS instances exhibit no determinism at all and thus twb and hwb match the size of the search space exactly). As the problems become bigger and their structure more complicated, however, the bound quality deteriorates. It should be interesting to perform this comparison on bigger problem instances, but as of now this is limited by the resources available in current computers.

6 CONCLUSIONS & FUTURE WORK

While asymptotic bounds for search algorithms can give a rough idea about problem hardness, it is often desirable to obtain a tighter, more fine-grained bound. As has previously been shown, this can be accomplished by looking at a suitable tree decomposition of the problem's underlying graph structure and the domains of variables in the decomposition clusters. This, however, is blind to determinism, which can greatly prune the search space in practice.

The contribution of this paper is to introduce ideas from the framework of hypertree decompositions into the bounding of the search space. This allows us to exploit determinism in the function specification, but only if it is beneficial to the overall complexity bound.

We demonstrated on a set of 112 belief networks and 30 weighted constraint networks that the proposed scheme is indeed able to further improve the bound on search complexity, in some cases by several orders of magnitude. On a subset of the instances we also showed that the bound can indeed be very tight, although it seems to deteriorate for bigger instances. In this respect we hope to be able to conduct more in-depth comparisons on even bigger problem instances in the future. Note that the ability to bound, sometimes accurately, the size of the search space in linear time is very important, especially for problem instances which are completely unsolvable exactly.

We believe that the current version of our bounding scheme can be further improved by incorporating some form of propagation of information down the bucket tree. Another path we plan to pursue is using approximate counting methods, such as sampling, to compute approximations to the number of solutions in each cluster, which will also approximate the number of nodes. Finally, for optimization tasks and for approximating branch-and-bound and best-first search algorithms we hope to accomplish further tightening of the search space using the cost function itself.

On a higher level, we plan to use the bounds we obtain to guide the selection of static and dynamic variable orderings. We also intend to deploy our scheme for parallelizing search algorithms over a networks of many machines (e.g., grids and clusters).

Acknowledgments

This work was partially supported by NSF grant IIS-0713118 and NIH grant R01-HG004175-02.

References

[Dechter and Pearl, 1989] R. Dechter and J. Pearl: Tree Clustering for Constraint Networks. *Artificial Intelligence* **38** (1989): 353–366.

[Dechter and Mateescu, 2007] R. Dechter and R. Mateescu: AND/OR search spaces for graphical models. In *Artificial Intelligence* **171** (2007): 73–106.

[Dechter et al., 2008] R. Dechter, L. Otten, and R. Marinescu: On the Practical Significance of Hypertree vs. Tree Width. In *Proceedings of ECAI'08*.

[Dermaku et al., 2005] A. Dermaku, T. Ganzow, G. Gottlob, B. McMahan, N. Musliu, M. Samer: Heuristic Methods for Hypertree Decompositions. *Technical Report DBAI-TR-2005-53*, Vienna University of Technology, 2005

[Fishelson et al., 2005] M. Fishelson, N. Dovgolevsky N, and D. Geiger: Maximum Likelihood Haplotyping for General Pedigrees. *Human Heredity* **59** (2005): 41–60.

[Gottlob et al., 2000] G. Gottlob, N. Leone, and F. Scarcello: A comparison of structural CSP decomposition methods. *Artificial Intelligence* **124** (2000): 243–282.

[Johnson, 1973] D. S. Johnson: Approximation algorithms for combinatorial problems. In *Proceedings of STOC'73*: 38–49.

[Kask et al., 2005] K. Kask, R. Dechter, J. Larrosa, and A. Dechter: Unifying tree decompositions for reasoning in graphical models. *Artificial Intelligence* **166** (2005): 165–193.

[Kjærulff, 1990] U. Kjærulff: Triangulation of Graphs – Algorithms Giving Small Total State Space. *Research Report R-90-09, Dept. of Mathematics and Computer Science*, Aalborg University 1990.

[Lauritzen and Spiegelhalter, 1988] S. L. Lauritzen and D. J. Spiegelhalter: Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society. Series B* **50(2)** (1988): 157–224.

[Mateescu and Dechter, 2005] R. Mateescu and R. Dechter: The Relationship Between AND/OR Search Spaces and Variable Elimination. In *Proceedings of UAI'05*: 380–387.

[Pearl, 1988] J. Pearl: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, 1988.

[Silberstein et al., 2006] M. Silberstein, A. Tzemach, N. Dovgolevskiy, M. Fishelson, A. Schuster, D. Geiger: Online system for faster linkage analysis via parallel execution on thousands of personal computers. *American Journal of Human Genetics*, 2006.

[Zabiyaka and Darwiche, 2006] Y. Zabiyaka and A. Darwiche: Bounding Complexity in the Presence of Functional Dependencies. In *Proceedings of SAT'06*: 116–129.

Observation Subset Selection as Local Compilation of Performance Profiles

Yan Radovilsky
Computer Science Dept.
Ben-Gurion University
84105 Beer-Sheva, Israel
yanr@cs.bgu.ac.il

Solomon Eyal Shimony
Computer Science Dept.
Ben-Gurion University
84105 Beer-Sheva, Israel
shimony@cs.bgu.ac.il

Abstract

Deciding what to sense is a crucial task, made harder by dependencies and by a non-additive utility function. We develop approximation algorithms for selecting an optimal set of measurements, under a dependency structure modeled by a tree-shaped *Bayesian network* (BN).

Our approach is a generalization of composing *anytime algorithm* represented by *conditional performance profiles*. This is done by relaxing the *input monotonicity* assumption, and extending the *local compilation* technique to more general classes of *performance profiles* (PPs). We apply the extended scheme to selecting a subset of measurements for choosing a maximum expectation variable in a binary valued BN, and for minimizing the worst variance in a Gaussian BN.

1 Introduction

A typical diagnostic system consists of two types of variables: tests (observable) and hypotheses (unobservable), with statistical dependencies among variables. Each test, if performed, consumes resources (time or money), and provides a measurement of one or more test variables. After obtaining the values of the selected tests, the distribution of the model is updated. An objective function specifies a reward given to the system for the posterior distribution. The system should make its selection so as to optimize the objective function, a hard problem in the general case. *Observation subset selection* (OSS) is a restricted version of this problem, where all measurements must be selected in advance, prior to any observations. In this paper we develop approximation algorithms for some settings of the OSS problem for tree-shaped dependency structures.

To tackle this problem we present OSS as a variant of the following well-known meta-reasoning problem. In systems composed of several *computational components* (CCs), the meta-level controller should reason about allocation of available computational resources for different CCs in order to optimize the overall performance of the entire system. This task is usually referred to in the research literature as the *meta-level resource allocation* (MRA) problem (see for example [11]). The standard approach used to optimize the MRA task was proposed by S. Zilberstein [10, 11, 12], the technique of *local compilation* (LC). This technique is applied to individual CCs, represented in a form of *conditional performance profiles*, and generates the optimal *time allocation scheme* for the entire system (see Section 2). However, local compilation requires the *input monotonicity* assumption and is, therefore, restricted to deterministic *performance profiles* with scalar output quality.

In this paper we relax the *input monotonicity* assumption and extend the LC technique to more general classes of PPs. We then apply the extended approximation scheme (Section 4) to find an approximate solution to the OSS problem, under two settings, both for dependencies modeled as a tree-shaped BN: a) finding a maximum expectation variable in a binary valued BN (Section 4.1), and b) minimizing the worst variance in a Gaussian BN (Section 4.2).

2 Background

Flexible computation refers to procedures that allow a graceful trade-off to be made between the quality of results and allocation of costly resources, such as time, memory, or information [3]. Since time is usually the main computational resource, there are several alternative terms used for reference to *flexible computation* in the research literature: *continual computation* [4, 5], *anytime computation* [8], and *anytime algorithms* [1, 10, 12].

To predict the quality of the result which depends on the amount of allocated time, a statistical model called a *performance profile* (PP) is employed. The most simple version of such a PP called an *expected performance profile* (EPP), a mapping from consumed time to an expected result quality, $Q : \mathbf{T} \rightarrow \mathbf{Q}$. Given *anytime algorithm* A described by EPP Q_A and time-dependent utility function $U : \mathbf{T} \times \mathbf{Q} \rightarrow \mathcal{R}$, optimal time allocation t^* can be derived as follows:

$$t^* = \arg\max_t U(t, Q_A(t)) \tag{1}$$

When an algorithm operates with some inexact inputs, the quality of its result may also strongly depend on the quality of the inputs. To address this point, a more flexible model, known as a *conditional performance profile* (CPP), is used. For example, an *anytime algorithm* with 2 inputs and one output can be represented by a CPP $Q : \mathbf{T} \times \mathbf{Q}^2 \rightarrow \mathbf{Q}$.

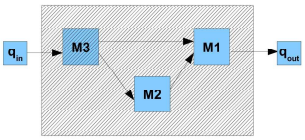


Figure 1: Composition graph

In a complex system, several CCs (or *anytime algorithms*) can be composed to achieve a required result. In this case, their PPs should be compiled in order to obtain an appropriate performance prediction for the entire system. Such a system is usually described in a graphical form by a *directed acyclic graph* (DAG), where each node corresponds to a single CC and is associated with an appropriate PP, while edges represent dependencies between input/output qualities of different CCs (Figure 1). This model is referred to as a *composition graph* (CG).

A *time allocation scheme* (TAS) $\hat{t} \in \mathbf{T}^n$ specifies allocation of time for each of n elementary CCs of a composite system. Output quality of the whole system can be expressed as a function of a TAS in a form of *composite expression* (CE) $\phi : \mathbf{T}^n \times \mathbf{Q}^k \rightarrow \mathbf{Q}$, where k is a number of external inputs. For example, the system in figure 1 has the following CE:

$$\phi(\hat{t}, q_{in}) = Q_1([\hat{t}]_1, Q_2([\hat{t}]_2, Q_3([\hat{t}]_3, q_{in})), Q_3([\hat{t}]_3, q_{in}))$$

The MRA problem for a *composite system* is to optimize a time-dependent utility function by selecting an appropriate TAS. Formally, given a *composite system* represented by CE ϕ, external input quality q_{in}, and utility function U, the goal is to find TAS $\hat{t}^* \in \mathbf{T}^n$ which maximizes the utility:

$$\hat{t}^* = \arg\max_{\hat{t} \in \mathbf{T}^n} U(\|\hat{t}\|, \phi(\hat{t}, q_{in})), \tag{2}$$

where operator $\|\cdot\|$ denotes a summation over all elements of its argument vector: $\|\hat{t}\| = \sum_i [\hat{t}]_i$.

An intuitive approach to tackle the MRA problem has been proposed in [10]. This approach, called *compilation of performance profiles*, involves construction of an appropriate PP for the entire system:

$$Q^c(t, q_{in}) = \max_{\hat{t} \in \mathbf{T}^n : \|\hat{t}\| = t} \phi(\hat{t}, q_{in}), \tag{3}$$

Each entry in Q^c is additionally associated with a corresponding TAS $\pi(t, q_{in}) = \hat{t}$ s.t. $\phi(\hat{t}, q_{in}) = Q^c(t, q_{in})$ and $\|\hat{t}\| = t$.

Once such a model is constructed, the optimal solution for the MRA problem can be derived from this model as $\hat{t}^* = \pi(t^*, q_{in})$, where t^* is the optimal total time allocation computed using equation 1.

Algorithm 1: method *RLC* for in-tree shaped composite systems

Input : s - a node in a composition graph.
Output: Composed PP for the subtree rooted in s.
if *(*$\mathtt{Pred}(s) = \emptyset$*)* **then**
    return Q_s;
foreach $s_i \in \mathtt{Pred}(s)$ **do**
    Let $Q^c_{s_i} \leftarrow \mathtt{RLC}(s_i)$;
Let $L \leftarrow \{Q^c_{s_i} : s_i \in \mathtt{Pred}(s)\}$;
return $\mathtt{Compose}(Q_s, L)$;

In general, the task of (global) *compilation* is computationally hard even for approximate solution [10]. However, for some restricted topologies (e.g. trees) S. Zilberstein [10] proposed an efficient algorithm based on the *local compilation* (LC) technique, summarized as RLC in Algorithm 1. In this technique a composite PP of each subtree is generated based on the CPP of its root and composed PPs of all its predecessors. Method *Compose* performs the basic composition operation, which results in the following composite PP:

$$Q^c_s(t, q_1, \ldots, q_k) = \max_{t_0, \ldots, t_k \in \mathbf{T} : \sum t_i = t} Q_s(t_0, Q^c_{s_1}(t_1, q_1), \ldots, Q^c_{s_k}(t_k, q_k)) \tag{4}$$

where $s_1, \ldots, s_k$ are the predecessors of node s.

For systems, where all CCs are represented by deterministic CPPs, RLC is proved to produce the correct result (equals to one obtained by *global compilation*) when the following assumptions hold:

- A *tree-structured* CG - each node has only one output which serves as an input for one successor node except for the root node, whose output is the resulting system output.

- *Input monotonicity* (IM) - the utility function and all involved CPPs are non-decreasing functions of input quality, i.e. $\forall q \geq p : Q(t,q) \geq Q(t,p)$.

Such restrictive assumptions are required to enable independent compilation of each subtree followed by *greedy* selection of one local TAS for each total allocation without harming optimality of the resulting solution. However, in practice, the IM assumption may be too restrictive. For example, a diagnostic system with a dependency model represented by Gaussian BN and a utility function depending on posterior marginal precision of several hypothesis nodes, violates this assumption. In this work we propose an extension to the CPP model, which allows relaxing the IM assumption.

Recent work exists in another line directly related to the OSS task. In [7] the authors proved that the OSS problem is NP^{PP}-hard even for tree-structured BNs. They proposed a polynomial time algorithm, based on *dynamic programming*, which constructs an optimal solution to a version of OSS restricted to chain topology, exact observation, and additive reward. In [9] a similar technique was used beyond the *exact observation* assumption, and determined a theoretical bound for the worst-case loss in expected reward. Our work can be seen as an extention of the latter research. Another approximation method based on greedy selection of test nodes is applicable when the reward function exhibits property of *sub-modularity* [6].

3 Generalized Local Compilation

This section generalizes the notion of a *performance profile* to a *reachable performance profile*, and adapts the LC technique to handle the extended model.

Definition 3.1 (Conditional performance)**.** *Conditional performance (CP) of a CC (either elementary or composite) is a tuple $(\hat{t}, p, q)$, where $\hat{t}$ is a local TAS (t-component), p is a required input quality (p-component), and q is an expected output quality, obtained by the CC when applied to inputs of quality p with TAS $\hat{t}$ (q-component).*

Either p or q may be represented by scalars or by vectors, while $\hat{t}$ is assumed to be a complete assignment of time allocations to all CCs of the system.

Definition 3.2 (Reachable performance profile)**.** *Reachable performance profile (RPP) of a CC is a set of CPs achievable by this CC.*

The RPP model can be derived from a CPP as follows:

$$R_A = \{(\pi_A(t,p), p, Q_A(t,p))\}, \quad (5)$$

where Q_A is a CPP, and R_A is an appropriate RPP. The converse is not always possible, because a RPP can contain more than one output quality for the same pair of total time allocation and input quality (each with a different *local* TAS).

We further extend our model by allowing backward conditioning, which means that output quality of a CC may depend on output quality of its successor. The general form of a CP in this case is $(\hat{t}, (p_{suc}, p_{pred}), (q_{suc}, q_{pred}))$, where the p-component has two parts: p_{suc} is a required output quality of the successor, and p_{pred} is a vector of required output qualities of all the predecessors; the q-component has two parts as well: an output quality towards its successor (q_{suc}), and a vector of qualities towards all its predecessors (q_{pred}). Such a form of CPs is useful in applications for BNs, where posterior probability distribution of a node (and its local reward) depends on the messages coming from all its neighbors (details appear in section 4).

In order to adapt the RLC algorithm to the extended settings, we need to modify the *Compose* method. This method is now applied to a list of profiles in the RPP representation, and its output should be in the RPP form as well. Moreover, the greedy selection reflected in the max operator in equation 3 strongly relies on the IM assumption. Since this assumption fails in the RPP case, we propose another approach. The composition process comprises two parts: the first part (*Construct*) considers all combinations of input CPs, and collects appropriate resulting CPs.

$$Q = \{(\hat{t}_0 + \hat{t}_1 + \ldots + \hat{t}_k, p_0, q_0) : (\hat{t}_i, p_i, q_i) \in Q_{s_i}, \\ (\hat{t}_0, (p_0, q_1, \ldots, q_k), (q_0, p_1, \ldots, p_k)) \in Q_s\}. \quad (6)$$

The second part, called *Purge* is applied to the set Q. *Purge* exploits *domination* and *equivalence* among reachable CPs in order to filter out irrelevant CPs.[1] The idea of pruning irrelevant CPs is very similar to one used in the *Incremental Pruning* algorithm [2] for filtering irrelevant α-vectors. The resulting RPP Q^c keeps one representative for each CP in Q:

$$\forall a \in Q \ \exists b \in Q^c \text{s.t. } (b \succ a) \vee (b \approx a).$$

where $\succ$ and $\approx$ are *domination* and *equivalence* operators respectively. These operators can be defined based on partial orders within each performance component. Assuming only a natural partial order in the t-component we obtain:

$$(\hat{t}, p, q) \approx (\hat{t}', p', q') \Leftrightarrow (\|\hat{t}\| = \|\hat{t}'\|) \wedge (p \overset{P}{\approx} p') \wedge (q \overset{Q}{\approx} q')$$

and

$$(\hat{t}, p, q) \succ (\hat{t}', p', q') \Leftrightarrow (\|\hat{t}\| < \|\hat{t}'\|) \wedge (p \overset{P}{\approx} p') \wedge (q \overset{Q}{\approx} q')$$

[1] The filtering can be applied after *Construct* terminates, but it is usually more efficient to perform filtering on-the-fly.

where $\overset{P}{\approx}$ and $\overset{Q}{\approx}$ are equivalence operators defined over the p and q components, respectively.

4 Observation Selection in BNs.

In this section we describe how our framework can be applied to OSS in BNs. We use the following notation:

- $\mathbf{X} = \{X_i : 1 \leq i \leq n\}$ - set of all state variables;
- $\mathbf{X}_H \subseteq \mathbf{X}$ - set of hypothesis state variables;
- $\mathbf{X}_M \subseteq \mathbf{X}$ - set of measurable state variables;
- $\mathbf{Y} = \{Y_i : X_i \in \mathbf{X}_M\}$ - set of test variables;
- N - BN over $\mathbf{X} \cup \mathbf{Y}$;
- $R : \Pr(\mathbf{X}_H) \rightarrow \mathcal{R}$ - reward function;
- $T(E) = \sum_{Y_i \in E} \tau_i$ - additive time consumption;
- B - time budget (maximum time for observation).

Definition 4.1 (OSS optimization problem)**.** *The OSS optimization problem is: given a tuple* (N, R, T, B)*, select a subset of observation variables* $E \subseteq \mathbf{Y}$ *which maximizes the expected reward:*

$$\hat{R}(\mathbf{X}_H|E) = \sum_{e \in Dom(E)} \Pr(E = e) R(\mathbf{X}_H|E = e) \quad (7)$$

subject to the budget limit $T(E) \leq B$*.*

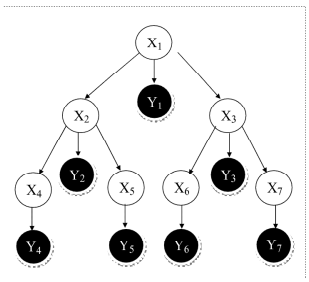


Figure 2: Out-tree BN topology.

We assume BNs with an out-tree shaped dependency graph rooted in node X_1 (as shown in figure 2). Let $\mathbf{X}_s^I$ denote a subset of $\mathbf{X}$, which consists of all nodes X_i in the subtree rooted in X_s (all descendants of X_s including X_s itself), and let $E \subseteq \mathbf{Y}$ be any given subset of observation ("evidence") nodes. We use the following notation to refer to other relevant subsets of nodes:

$$\begin{aligned}
&\mathbf{X}_s^O = \mathbf{X} \setminus \mathbf{X}_s^I;\\
&\mathbf{Y}_s^I = \{Y_i : X_i \in (\mathbf{X}_M \cap \mathbf{X}_s^I)\};\\
&\mathbf{Y}_s^O = \mathbf{Y} \setminus \mathbf{Y}_s^I;\\
&E_s^I = E \cap \mathbf{Y}_s^I;\\
&E_s^O = E \cap \mathbf{Y}_s^O.
\end{aligned}$$

4.1 OSS in discrete BNs

We now apply our approach to OSS in tree-shaped BNs with discrete variables. Since solving this problem for the general setting (even for the tree-structured topology) is proved to be NP^{PP}-hard [7], we restrict our focus to BNs with boolean variables ($Dom(X_i) = Dom(Y_i) = \{0, 1\}$), and consider the following reward function, defined for an arbitrary set of nodes A:

$$R(A|E = e) = \max_{X_i \in A} R_i(X_i|E = e),$$

where each $R_i : \Pr(X_i) \rightarrow \mathcal{R}$ is a local reward function:

$$R_i(X_i|E = e) = \begin{cases} \Pr(X_i = 1|E = e) & : \text{if}(X_i \in X_H), \\ 0 & : \text{otherwise.} \end{cases} \quad (8)$$

We refer to this version of OSS as *Boolean OSS* (BOSS).

The BN can be specified by the following parameters:

$$\alpha_i = \begin{cases} \Pr(X_i = 1) & : \text{if } X_i \text{ is a root,} \\ \Pr(X_i = 1|X_{Prev(i)} = 1) & : \text{otherwise;} \end{cases}$$

$$\beta_i = \begin{cases} \Pr(X_i = 1) & : \text{if } X_i \text{ is a root,} \\ \Pr(X_i = 1|X_{Prev(i)} = 0) & : \text{otherwise;} \end{cases}$$

$$\theta_i = \begin{cases} \Pr(Y_i = 0|X_i = 1) & : \text{if}(X_i \in \mathbf{X}_M), \\ 1 & : \text{otherwise;} \end{cases}$$

$$\zeta_i = \begin{cases} \Pr(Y_i = 1|X_i = 0) & : \text{if}(X_i \in \mathbf{X}_M), \\ 0 & : \text{otherwise;} \end{cases}$$

We make the following simplifying assumptions about the involved observation process:

1. Probability of a *false positive* observation result for all nodes is bounded by a small constant ζ_{max} (that is $\forall i : \zeta_i \leq \zeta_{max}$);
2. Only hypothesis nodes have directly attached observations ($\mathbf{X}_M \subseteq \mathbf{X}_H$).

Henceforth this set of assumptions is called the *restricted false positive* property. In the extreme case ($\zeta_{max} = 0$) we get a *false-positive-free* observation process. Despite the relatively restricted setting, we have the following complexity result:

Theorem 1 (Hardness of BOSS)**.** *Finding an exact solution to the BOSS problem is NP-hard even when all state variables are independent ($\alpha_i = \beta_i$) and all observations are exact ($\theta_i = \zeta_i = 0$).*

Proof is by reduction from *Knapsack*, which is a well-known *NP*-complete problem. Below, we show how the BOSS problem can be reduced to a special case of MRA and then solved (approximately) by the RLC algorithm.

In order to apply the RLC algorithm we must specify the problem in terms of a composite system. Deriving the corresponding CG is straightforward: the graph has in-tree form and can be obtained from the dependency graph by simply reversing directions of all arcs.

Careful inspection of the *false-positive-free* property yields that observing one positive value at any observation node ($Y_i = 1$) provides a sufficient evidence for determining the reward value ($R(\mathbf{X}_H|Y_i = 1, E = e) = R_i(X_i|Y_i = 1) = 1$), regardless of other observations. We employ this fact to obtain a recursive decomposition of the expected reward.

We specify output quality (q-component) of exploring subset $E \subseteq \mathbf{Y}$ w.r.t. subtree $\mathbf{X}_s^I$ as the triple (f, g, r):

$$\begin{aligned} f &= \Pr(\hat{e}_s^I|X_s = 1), \\ g &= \Pr(\hat{e}_s^I|X_s = 0), \\ r &= R(\mathbf{X}_s^I|\hat{e}), \end{aligned}$$

where $\hat{e}$, $\hat{e}_s^I$, and $\hat{e}_s^O$ denote assignments of all zeros to E, E_s^I, and E_s^O respectively. While f and g components depend only on observations inside the subtree ($\hat{e}_s^I$), to determine the value of the r-component we need additional information from outside the subtree, provided by the p-component: $p = \Pr(X_s = 1|\hat{e}_s^O)$.

For each quality component we define one corresponding domain set (of relevant values):

$$\begin{aligned} \mathbf{P}_s &= \left\{\Pr(X_s = 1|E_s^O = \hat{e}_s^O) : E \subseteq \mathbf{Y}\right\} \\ \mathbf{F}_s &= \left\{\Pr(E_s^I = \hat{e}_s^I|X_s = 1) : E \subseteq \mathbf{Y}\right\} \\ \mathbf{G}_s &= \left\{\Pr(E_s^I = \hat{e}_s^I|X_s = 0) : E \subseteq \mathbf{Y}\right\} \\ \mathbf{R}_s &= \left\{R(\mathbf{X}_s^I|E = \hat{e}) : E \subseteq \mathbf{Y}\right\}. \end{aligned}$$

We also define combined domain sets $\mathbf{Q}_s = \mathbf{F}_s \times \mathbf{G}_s \times \mathbf{R}_s$. Finally, all alternative assignments to a number of observations (measurements) in node X_s is represented by set $\mathbf{M}_s$. In the basic OSS setting we have at most one observation per node:

$$\mathbf{M}_s = \begin{cases} \{0, 1\} & : \text{if } (X_s \in \mathbf{X}_M), \\ \{0\} & : \text{otherwise.} \end{cases}$$

However, the model can be easily extended to multiple observations (by specifying appropriate $\mathbf{M}_s$ sets).

RPPs of observation nodes contain CPs with no condition (denoted by $\emptyset$ in the p-component):

$$Q_s^Y = \{(m\hat{\tau}_s, \emptyset, m) : m \in \mathbf{M}_s\}, \tag{9}$$

where $\hat{\tau}_s$ is an assignment of τ_s time units to s and 0 to all the other CCs.

All leaf X-nodes are associated with RPPs of the following form:

$$Q_s^X = \{(\hat{0}, u, \psi_s(u)) : u \in \mathbf{P}_s \times \mathbf{M}_s\} \tag{10}$$

where $\hat{0}$ denotes a zero time allocation (to all CCs), and $\psi_s : \mathbf{P}_s \times \mathbf{M}_s \to \mathbf{Q}_s$ is a vector function defined as follows:

$$\begin{aligned} &\psi_s(p, m) = (f, g, r), \\ &f = \Pr(\hat{e}_s^I|X_s = 1) = \theta_s^m, \\ &g = \Pr(\hat{e}_s^I|X_s = 0) = (1 - \zeta_s)^m, \\ &r = R_s(X_s|\hat{e}) = \begin{cases} \frac{pf}{L(f,g,p)} & : \text{if } (X_s \in \mathbf{X}_H), \\ 0 & : \text{otherwise} \end{cases} \end{aligned} \tag{11}$$

In our notation $L(\cdot, \cdot, \cdot)$ stands for the operator of linear interpolation defined as follows:

$$L(a, b, c) = ca + (1 - c)b. \tag{12}$$

RPP of any non-leaf node X_s with k children $(X_{s_1}, \ldots, X_{s_k})$ is specified as follows:

$$Q_s^X = \{(\hat{0}, u, \psi_s(u)) : u \in \mathbf{P}_s \times \mathbf{M}_s \times \mathbf{Q}_{s_1} \times \cdots \times \mathbf{Q}_{s_k}\} \tag{13}$$

where
$\psi_s : \mathbf{P}_s \times \mathbf{M}_s \times \mathbf{Q}_{s_1} \times \cdots \times \mathbf{Q}_{s_k} \to \mathbf{Q}_s \times \mathbf{P}_{s_1} \times \cdots \times \mathbf{P}_{s_k}$ is a vector function defined as follows:

$$\begin{aligned} &\psi_s(p, m, (f_1, g_1, r_1), \ldots, (f_k, g_k, r_k)) = \\ &\quad = ((f, g, r), p_1, \ldots, p_k), \\ &r = \max\{r_0, r_1, \ldots, r_k\}, \\ &r_0 = R_s(X_s|\hat{e}) = \begin{cases} \frac{pf}{L(f,g,p)} & : \text{if } (X_s \in \mathbf{X}_H), \\ 0 & : \text{otherwise} \end{cases} \\ &f = \Pr(\hat{e}_s^I|X_s = 1) = \theta_s^m \prod_{1 \le i \le k} f_i', \\ &g = \Pr(\hat{e}_s^I|X_s = 0) = (1 - \zeta_s)^m \prod_{1 \le i \le k} g_i', \\ &p_i = \Pr(X_{s_i} = 1|\hat{e}_{s_i}^O) = \\ &\quad = \frac{L(\alpha_{s_i}, \beta_{s_i}, p\theta_s^m \prod_{j \ne i} f_j')}{L(\theta_s^m \prod_{j \ne i} f_j', (1 - \zeta_s)^m \prod_{j \ne i} g_j', p)}, \\ &f_i' = \Pr(\hat{e}_{s_i}^I|X_s = 1) = L(f_i, g_i, \alpha_{s_i}), \\ &g_i' = \Pr(\hat{e}_{s_i}^I|X_s = 0) = L(f_i, g_i, \beta_{s_i}). \end{aligned} \tag{14}$$

While all Q_s^Y RPPs can be specified explicitly (as a list of CPs), the Q_s^X RPPs generally cannot, due to a possibly exponential size of domains $\mathbf{P}_s$, $\mathbf{F}_s$, $\mathbf{G}_s$ and $\mathbf{R}_s$ in their input condition. Instead, we represent them in an implicit form by providing (parameters of) the involved ψ_s functions.

During the compilation process (according to the RLC algorithm) method *Compose* is applied to lists of RPPs. At each application one composed RPP Q_s^c (which represents the whole subtree $\mathbf{X}_s^I$) is generated. Due to backward conditioning on elements of $\mathbf{P}_s$ which can contain exponential number of elements), such a RPP cannot be derived exactly (and explicitly). There is also no obvious way to specify it implicitly by providing some predefined functions (as we do in case of individual RPPs). To address this problem, we propose to approximate all $\mathbf{P}_s$ sets by one uniform grid $\mathbf{D}_p$ defined as follows:

$$\mathbf{D}_p = \left\{ \left(k + \frac{1}{2} \right) \epsilon_p : k \in 0, \ldots, d_p - 1 \right\} \tag{15}$$

where $0 < \epsilon_p < 1$ is a small constant, and $d_p = \left\lceil \frac{1}{\epsilon_p} \right\rceil$ is a number of intervals of size ϵ_p in the range $[0, 1]$. Thus, set $\mathbf{D}_p$ has a fixed number of elements (d_p) which makes efficient enumeration possible.

We define discretization function λ_p, which maps any value $p \in \mathbf{P}_s$ to an appropriate point in set $\mathbf{D}_p$:

$$\lambda_p(p) = \left(\left\lfloor \frac{p}{\epsilon_p} \right\rfloor + \frac{1}{2} \right) \epsilon_p \tag{16}$$

This discretization induces approximate equivalence relation $\overset{P}{\approx}$ among elements of $\mathbf{P}_s$ defined as follows:

$$p \overset{P}{\approx} p' \Leftrightarrow \lambda_p(p) = \lambda_p(p') \tag{17}$$

We apply a similar discretization technique to other domain sets ($\mathbf{F}_s$, $\mathbf{G}_s$, and $\mathbf{R}_s$) with discretization steps ϵ_f, ϵ_g and ϵ_r respectively. The appropriate grids ($\mathbf{D}_f$, $\mathbf{D}_g$, $\mathbf{D}_r$), discretization functions (λ_f, λ_g, λ_r) and *equivalence* operators ($\overset{F}{\approx}$, $\overset{G}{\approx}$, $\overset{R}{\approx}$) are defined accordingly. The composed *equivalence* operator $\overset{Q}{\approx}$ is defined as follows:

$$(f, g, r) \overset{Q}{\approx} (f', g', r') \Leftrightarrow (f \overset{F}{\approx} f') \wedge (g \overset{G}{\approx} g') \wedge (r \overset{R}{\approx} r')$$

To complete the specification, we define the following comprehensive utility function:

$$U(\hat{t}, p, (f, g, r)) = \tag{18}$$
$$= \begin{cases} L(r, 1, L(f, g, \alpha_1)) & : \text{if } (p \overset{P}{\approx} \alpha_1 \wedge \|\hat{t}\| \leq B), \\ -\infty & : \text{otherwise,} \end{cases}$$

After compiling RPPs of the entire tree, the resulting RPP Q_1^c can be used to select a near-optimal TAS w.r.t. this utility function. Let E^* denote the optimal observation subset, and let E be a subset corresponding to the TAS selected based on the resulting RPP.

Theorem 2 (Approximation quality of RLC for BOSS)**.** *The RLC algorithm applied to a transformed instance of BOSS with out-tree topology and a restricted false positive observation process approximates the optimal solution within additive factor of* Δ_u*, which is bounded as follows:*

$$\Delta_u = \hat{R}(\mathbf{X}|E^*) - \hat{R}(\mathbf{X}|E) \leq h\epsilon_p + 2n\epsilon_s + \epsilon_r + n\zeta_{max} \tag{19}$$

where h *is a height of the tree, and* $\epsilon_s = \max\{\epsilon_f, \epsilon_g\}$.

Selection of the appropriate values for the grid steps depends on the required precision of the solution. To ensure approximation with $\Delta_u \leq \epsilon + n\zeta_{max}$ (in worst case) we can select $\epsilon_p = \frac{\epsilon}{3h}$, $\epsilon_f = \epsilon_g = \frac{\epsilon}{6n}$, and $\epsilon_r = \frac{\epsilon}{3}$.

Due to monotonicity in total time, any composed RPP can be represented by the 4-dimensional table ($\mathbf{D}_p \times \mathbf{D}_f \times \mathbf{D}_g \times \mathbf{D}_r$), where each entry is associated with at most one appropriate partial TAS. The number of entries (CPs) in such a table is bounded as follows:

$$|Q_s^c| \leq d_p d_f d_g d_r = \left\lceil \frac{3h}{\epsilon} \right\rceil \left\lceil \frac{6n}{\epsilon} \right\rceil^2 \left\lceil \frac{3}{\epsilon} \right\rceil. \tag{20}$$

Worst-case complexity for the complete run of RLC is $O\left(\frac{n^4}{\epsilon^4}\right)$ time, $O\left(\frac{n^3}{\epsilon^4}\right)$ space for a chain topology, and $O\left(\frac{n^{2c+1}hc}{\epsilon^{2c+2}}\right)$ time, $O\left(\frac{n^{2c}h^2c}{\epsilon^{2c+2}}\right)$ space for a tree with a maximum branching factor of c.

4.2 OSS in Gaussian Bayesian networks

Gaussian Bayesian network (GBN) is a special case of BN, where the *conditional probability distributions* of the variables are Normal (Gaussian) distributions:

$$X_i | Pa(X_i) \sim N\left(\mu_i + \sum_{X_j \in Pa(X_i)} a_{i,j}(X_j - \mu_j), \sigma_i^2 \right)$$

We parametrize our GBN model as follows:

$$\alpha_i = a_{i,Prev(i)}^2,$$
$$\beta_i = Prec(X_i | X_{Prev(i)}) = \frac{1}{\sigma_i^2},$$
$$\theta_i = \begin{cases} Prec(Y_i|X_i) & : \text{if } (X_i \in \mathbf{X}_M), \\ 0 & : \text{otherwise.} \end{cases}$$

In our notation $Prec(\cdot)$ denotes the precision operator, which is reciprocal to variance:

$$Prec(X_i|E) = \frac{1}{Var(X_i|E)}. \tag{21}$$

For *Gaussian OSS* (GOSS) we consider the following reward function:

$$R(A|E=e) = \min_{X_i \in A} R_i(X_i|E),$$

where $R_i : \Pr(X_i) \to \mathcal{R}$ are local reward functions:

$$R_i(X_i|E) = \begin{cases} \overline{\log}_{a,b}(Prec(X_i|E)) & : \text{if}(X_i \in \mathbf{X}_H), \\ 1 & : \text{otherwise.} \end{cases} \tag{22}$$

Here a and b are two parameters that determine a range of distinguishable (for reward) values of precision, and $\overline{\log}_{a,b}(\cdot)$ denotes a normalized (and truncated at its extreme points) log operator, defined as follows:

$$\overline{\log}_{a,b}(p) = \begin{cases} 0 & : \text{if } (p \le a), \\ 1 & : \text{if } (p \ge b), \\ \frac{\log p - \log a}{\log b - \log a} & : \text{otherwise.} \end{cases} \tag{23}$$

Theorem 3 (Hardness of GOSS)**.** *Finding an exact solution for a general instance of the GOSS problem is NP-hard even for a tree-shaped GBN.*

Proof is by reduction from *Knapsack*. A polynomial scheme for approximate solution of GOSS, similar to one presented for BOSS, follows.

To apply the RLC algorithm we specify the problem in terms of a composite system. The CG is as for BOSS. All domain sets (except for $\mathbf{M}_s$, that remain the same) should be redefined as follows:

$$\mathbf{P}_s = \{Prec(X_s|E_s^O) : E \subseteq \mathbf{Y}\}, \tag{24}$$
$$\mathbf{F}_s = \{Prec(X_s|E_s^I) : E \subseteq \mathbf{Y}\}, \tag{25}$$
$$\mathbf{R}_s = \{R_s(X_s|E) : E \subseteq \mathbf{Y}\}, \tag{26}$$
$$\mathbf{Q}_s = \mathbf{F}_s \times \mathbf{R}_s. \tag{27}$$

We need to reformulate, in the definition of RPPs, the specification of the ψ_s functions. For the leaf nodes ψ_s is defined as follows:

$$\psi_s(p, m) = (f, r), \tag{28}$$
$$f = Prec(X_s|E_s^I) = m\theta_s,$$
$$r = R_s(X_s|E) = \begin{cases} \overline{\log}_{a,b}(p+f) & : \text{if } (X_s \in \mathbf{X}_H), \\ 1 & : \text{otherwise,} \end{cases}$$

For the non-leaf nodes we have:

$$\psi_s(p, m, (f_1, r_1), \ldots, (f_k, r_k)) = ((f, r), p_1, \ldots, p_k);$$
$$f = Prec(X_s|E_s^I) = m\theta_s + \sum_{1 \le i \le k} f_i',$$
$$r = \min\{r_0, r_1, \ldots, r_k\},$$
$$r_0 = R_s(X_s|E) = \begin{cases} \overline{\log}_{a,b}(p+f) & : \text{if } (X_s \in \mathbf{X}_H), \\ 1 & : \text{otherwise,} \end{cases}$$
$$p_i = Prec(X_{s_i}|E_{s_i}^O) = J(\beta_{s_i}, \alpha_{s_i}(m\theta_s + p + \sum_{j \ne i} f_j')),$$
$$f_i' = Prec(X_s|E_{s_i}^I) = \frac{J(f_i, \beta_{s_i})}{\alpha_{s_i}}, \tag{29}$$
$$\tag{30}$$

In our notation $J(\cdot,\cdot)$ stands for the operator of precision propagation defined as follows:

$$J(a, b) = \begin{cases} 0 & : \text{if } (a = b = 0), \\ \frac{ab}{a+b} & : \text{otherwise.} \end{cases} \tag{31}$$

As in case of BOSS, to prevent exponential growth of the composed RPPs we apply discretization to all domains by appropriate grids. Grid $\mathbf{D}_r$ and corresponding discretization function λ_r are defined exactly as in BOSS. To define $\mathbf{D}_p$ we use its projection $\mathbf{D}_p'$ to the $[0, 1]$ interval ($\mathbf{D}_p'$ is defined exactly as $\mathbf{D}_p$ in the BOSS case):

$$\mathbf{D}_p = \{p : \overline{\log}_{a,b}(p) \in \mathbf{D}_p'\}, \tag{32}$$

We express the discretization function λ_p through its projected version λ_p' (which is defined as λ_p in BOSS):

$$\lambda_p(p) = \lambda_p'(\overline{\log}_{a,b}(p)). \tag{33}$$

Grid $\mathbf{D}_f$ and the corresponding discretization function λ_f are similarly defined.

Equivalence operators $\overset{P}{\approx}$, $\overset{F}{\approx}$, and $\overset{R}{\approx}$ are defined as in BOSS. The composed *equivalence* operator $\overset{Q}{\approx}$ is:

$$(f, r) \overset{Q}{\approx} (f', r') \Leftrightarrow (f \overset{F}{\approx} f') \wedge (r \overset{R}{\approx} r')$$

The comprehensive utility function is as follows:

$$U(\hat{t}, p, (f, r)) = \begin{cases} r & : \text{if } ((p \overset{P}{\approx} \beta_1) \wedge (\|\hat{t}\| \le B)), \\ -\infty & : \text{otherwise} \end{cases} \tag{34}$$

After compiling the composite system (using the RLC algorithm), a near-optimal TAS can be selected from the resulting RPP Q_1^c w.r.t. this utility function. Let E^* denote an optimal observation subset, and E a subset corresponding to the TAS selected from Q_1^c.

Theorem 4 (Approximation quality of RLC for GOSS)**.** *The RLC algorithm applied to a transformed instance of GOSS problem with out-tree topology approximates the optimal solution E^* within additive factor of Δ_u, bounded as follows:*

$$\Delta_u = \hat{R}(\mathbf{X}_H|E^*) - \hat{R}(\mathbf{X}_H|E) \leq h\epsilon_p + h\epsilon_f + \epsilon_r \quad (35)$$

To ensure approximation with $\Delta_u \leq \epsilon$ (in worst case) we can select $\epsilon_p = \epsilon_f = \frac{\epsilon}{h}$, and $\epsilon_r = \epsilon$.

Any composed RPP Q_s^c can be represented by a 3-dimensional ($\mathbf{D}_p \times \mathbf{D}_f \times \mathbf{D}_r$) table with a number of entries bounded as follows:

$$|Q_s^c| \leq d_p d_f d_r = \left\lceil \frac{h}{\epsilon} \right\rceil^2 \left\lceil \frac{1}{\epsilon} \right\rceil . \quad (36)$$

The appropriate worst-case complexity for the complete run of the RLC algorithm is $O\left(\frac{nh^2}{\epsilon^2}\right)$ time, $O\left(\frac{h^2}{\epsilon^3}\right)$ space for a chain topology, and $O\left(\frac{nh^{c+1}c}{\epsilon^{c+2}}\right)$ time, $O\left(\frac{h^{c+2}c}{\epsilon^{c+2}}\right)$ space for a tree with maximum branching factor of c.

5 Summary

In this paper we extended the concept of CPP, and presented an efficient technique for compiling a composite system beyond the *input monotonicity* assumption. The extended scheme has been applied to optimizing a set of measurements in two different settings (for choosing a maximum expectation variable in a binary valued BN, and for minimizing the worst variance in a Gaussian BN). Polynomial time methods have been presented for both problems, and quality of approximation has been theoretically determined.

Applying our framework to real-world domains as an empirical evaluation is underway. Further extending the framework to deal with more general system topologies, tractable strategies for active monitoring are possible directions for future research.

Acknowledgements

Partially supported by the IMG4 consortium (under the Ministry of Industry, Trade and Labor of Israel MAGNET program), by the Lynn and William Frankel Center for Computer Sciences, and by the Paul Ivanier Center for Robotics.

References

[1] M. Boddy and T. Dean. Solving time-dependent planning problems. In *IJCAI*, pages 979–984, 1989.

[2] A. Cassandra, M. L. Littman, and N. L. Zhang. Incremental Pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of (UAI–97)*, pages 54–61, 1997.

[3] E. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *UAI*, pages 301–324, 1987.

[4] E. Horvitz. Models of continual computation. In *AAAI/IAAI*, pages 286–293, 1997.

[5] E. Horvitz. Continual computation policies for allocating offline and real-time resources. In *IJCAI*, pages 1280–1287, 1999.

[6] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *UAI*, pages 324–331, 2005.

[7] A. Krause and C. Guestrin. Optimal nonmyopic value of information in graphical models - efficient algorithms and theoretical limits. In *IJCAI*, pages 1339–1345, 2005.

[8] A. I. Mouaddib and S. Zilberstein. Knowledge-based anytime computation. In *IJCAI*, pages 775–783, 1995.

[9] Y. Radovilsky, G. Shattah, and E. S. Shimony. Efficient deterministic approximation algorithm for nonmyopic value of information in graphical models. In *SMC Conference*, Taipei, Taiwan, 2006.

[10] S. Zilberstein. Operational rationality through compilation of anytime algorithms. Technical report, Computer Science Division, University of California at Berkeley, 1993. PhD Dissertation.

[11] S. Zilberstein. Optimizing decision quality with contract algorithms. In *IJCAI*, pages 1576–1582, 1995.

[12] S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.

Improving the Accuracy and Efficiency of MAP Inference for Markov Logic

Sebastian Riedel
Institute for Collaborating and Communicating Systems
School of Informatics
University of Edinburgh
EH81BL Edinburgh, Scotland

Abstract

In this work we present Cutting Plane Inference (CPI), a Maximum A Posteriori (MAP) inference method for Statistical Relational Learning. Framed in terms of Markov Logic and inspired by the Cutting Plane Method, it can be seen as a meta algorithm that instantiates small parts of a large and complex Markov Network and then solves these using a conventional MAP method. We evaluate CPI on two tasks, Semantic Role Labelling and Joint Entity Resolution, while plugging in two different MAP inference methods: the current method of choice for MAP inference in Markov Logic, MaxWalkSAT, and Integer Linear Programming. We observe that when used with CPI both methods are significantly faster than when used alone. In addition, CPI improves the accuracy of MaxWalkSAT and maintains the exactness of Integer Linear Programming.

1 INTRODUCTION

Many tasks in Machine Learning are inherently relational: the label given to an object often depends on labels given to a set of related objects. For example, in Semantic Role Labelling [Carreras and Marquez, 2005] we are asked to label phrases with the role they play with respect to a given verb. Here the role given to one phrase depends on roles we have assigned to other phrases in the same sentence. It is, for instance, not possible to have two phrases both labelled as the agent of the same verb.

Statistical Relational Learning [SRL, Ng and Subrahmanian, 1992, Koller, 1999] seeks to provide generic, solid and efficient means to solve such relational tasks. It typically uses variants of First Order Logic to describe Graphical Models with repetitive structure in a compact fashion. This has two main benefits. Firstly, the meta-information provided by the first order model can be used to avoid a full instantiation of the Graphical Model in inference and learning. This can yield faster runtime and higher accuracy [Koller, 1999, de Salvo Braz et al., 2005, Singla and Domingos, 2006b]. Secondly, an SRL language along with a powerful interpreter allows application developers to focus on models, and machine learning researchers to focus on foundations. This paradigm of decoupling applications and algorithms has increased the speed of development in many domains [Domingos, 2006].

Markov Logic [ML, Richardson and Domingos, 2005] is an expressive SRL language that combines First Order Logic and Markov Networks. It has been successfully used for several tasks such as Information Extraction [Poon and Domingos, 2007] and Entity Resolution [Singla and Domingos, 2006a].

For most Markov Logic applications we need to solve the Maximum A Posteriori (MAP) problem of finding the most likely solution given some observation. Richardson and Domingos [2005] proposed the use of MaxWalkSAT [MWS, Kautz et al., 1996] to tackle this problem. In our experiments we apply MWS to two rather simple ML models, one for Semantic Role Labelling and one for Joint Entity Resolution. Here we found MWS to be both slow and inaccurate. However, before languages like ML can ever be used to solve tasks like joint inference in large Natural Language Processing applications [Domingos, 2007] we surely need to able to efficiently and accurately solve comparatively simple ones.

Rather than investigating the use of other solvers such as Belief Propagation and its variants, which can perform quite poorly for the large, densely connected and partially deterministic networks we encounter, we focus on tackling this problem by introducing a *meta* algorithm: *Cutting Plane Inference* (CPI) inspired by the Cutting Plane Method [Dantzig et al., 1954].

CPI incrementally instantiates only those portions of the complete Markov Network for which a current solution can be further optimised and solves these using

an existing inference method. Often these partial problems are significantly smaller and less complex. Consequently, they are more easily solved than the complete problem.

Empirically we show that for Semantic Role Labelling CPI plus MWS is significantly faster and more accurate than MWS alone. When used with Integer Linear Programming (ILP), CPI achieves optimal accuracy due to the exactness of ILP, yet runs significantly faster than when using ILP alone. With this accuracy we are able to achieve state-of-the-art results in Semantic Role Labelling with minimal engineering effort. When tested on an Joint Entity Resolution model taken from from the Markov Logic literature [Singla and Domingos, 2005] CPI with MWS does again better than MWS alone both in terms of speed and accuracy. CPI with ILP allows us to perform efficient and *exact* inference on this task while ILP alone is infeasible.

In the next section of this paper we will present Markov Logic. Section 3 shows two ways of solving the MAP problem for the Markov Networks that Markov Logic describes: MWS and ILP. In section 4 Cutting Plane Inference is presented and we formally show how the accuracy of CPI depends on the accuracy of the base solver. Section 5 compares CPI in combination with MWS and ILP to plain MWS and ILP on two tasks. The first is Semantic Role Labelling, the second Joint Entity Resolution. We conclude with section 6.

2 MARKOV LOGIC

Markov Logic [ML, Richardson and Domingos, 2005] is an SRL language based on First Order Logic and Markov Networks. It can be seen as a formalism that extends First Order Logic to allow formulae that can be violated with some penalty. From an alternative point of view, it is a expressive template language that uses First Order Logic formulae to instantiate Markov Networks of repetitive structure. [1]

Let us introduce some notation by example. Assume a simplified version of Semantic Role Labelling where we use an unary predicate *agent* to select the constituent that acts as agent for a given verb. We also maintain a set of additional predicates that provide information about constituents and their relation to the verb. For example, *left* can be a unary predicate that denotes constituents to the left of the verb. The set of all predicates will be called P. We also maintain a *finite* set C of constants representing constituents, words, tags etc.

In the following we will use n_p to denote the arity of a predicate p, and thus $n_{left} = n_{agent} = 1$. In formulae we will denote logical variables using the letter v and some subscript such as v_1. For example, in

$$\phi_1 : agent\left(v_1\right) \Rightarrow left\left(v_1\right)$$

v_1 is a variable and in

$$\phi_2 : v_1 \neq v_2 \wedge agent\left(v_1\right) \Rightarrow \neg agent\left(v_2\right)$$

v_1 and v_2 are variables. The number of free variables of a formula ϕ will be denoted with n_ϕ, thus $n_{\phi_1} = 1$ and $n_{\phi_2} = 2$. A *grounding* $\phi\left[v_1/c_1, \ldots, v_{n_\phi}/c_{n_\phi}\right]$ is generated by replacing each occurrence of each v_i with the constant c_i. We will often write $\phi\left[\mathbf{v}/\mathbf{c}\right]$ to mean $\phi\left[v_1/c_1, \ldots, v_{n_\phi}/c_{n_\phi}\right]$. For example, $\phi_2\left[\mathbf{v}/\mathbf{c}\right] = c_1 \neq c_2 \wedge agent\left(c_1\right) \Rightarrow \neg agent\left(c_2\right)$.

A formula that does not contain any variables is *ground*. A formula that contains a single predicate and nothing else is an *atom*. A set of ground atoms is called a *possible world* [Genesereth and Nilsson, 1987]. We say that a possible world W *satisfies a formula* ϕ and write $\vDash_W \phi$ if ϕ is true in W. For example, the possible world $\{agent\left(c_1\right)\}$ does not satisfy $\phi_1\left[v_1/c_1\right]$; the possible world $\{left\left(c_1\right), agent\left(c_1\right)\}$ does. In the following we will identify the binary vector $\mathbf{y} = \left(y_{p(\mathbf{c})}\right)_{p \in P, \mathbf{c} \in C^{n_p}}$ with the possible world $\left\{p\left(\mathbf{c}\right) | y_{p(\mathbf{c})} = 1\right\}$. For the sake of brevity we will often write $y^p_{\mathbf{c}}$ instead of $y_{p(\mathbf{c})}$. The set of all possible worlds we can construct using a set of predicates P and a set of constants C is $\mathcal{Y}_{P,C}$.

In First Order Logic a knowledge base is a set of formulae. It describes the set of possible worlds that for which all its formulae are satisfied. In Markov Logic the equivalent of a first order knowledge base is a *Markov Logic Network* (MLN). Instead of classifying models as either consistent (all formulae are satisfied) or inconsistent (some are not) an MLN maps each possible world to a probability. This allows us to model uncertainty in our beliefs about the world. For example, a world in which $agent\left(c_1\right) \Rightarrow left\left(c_1\right)$ does not hold should not be impossible, it should just be a bit less likely because the agent of a verb tends to be on its left but can appear on its right in passive constructions.

We define an MLN M as set of pairs $\{(\phi_i, w_i)\}_i$ where each ϕ_i is a formula in First Order Logic and $w_i \in \mathbb{R}$ is a real number. Together with a *finite* set of constants C, an MLN M defines a log-linear probability distribution over possible worlds $\mathbf{y} \in \mathcal{Y}_{P,C}$ as follows

$$p\left(\mathbf{y}\right) = \frac{1}{Z} \exp\left(\sum_{(\phi,w) \in M} w \sum_{\mathbf{c} \in C^{n_\phi}} f^\phi_{\mathbf{c}}\left(\mathbf{y}\right)\right) \quad (1)$$

where the feature function $f^\phi_{\mathbf{c}}$ is defined as

$$f^\phi_{\mathbf{c}}\left(\mathbf{y}\right) = \mathbb{I}\left(\vDash_{\mathbf{y}} \phi\left[v_1/c_1, \ldots, v_{n_\phi}/c_{n_\phi}\right]\right)$$

[1] Note that while this paper focuses on Markov Logic due to its expressive power and possibility of undirected dependencies, we note that much of the work reported here can be transferred to other formalisms.

Z is a normalisation constant, $\mathbb{I}(true) = 1$ and $\mathbb{I}(false) = 0$.

This distribution is strictly positive and corresponds to a Markov Network which is referred to as the *Ground Markov Network*. Note that we can represent hard constraints using very large weights.

For example, with $M = \{(\phi_1, 2.5), (\phi_2, 1.2)\}$ and the finite set of constants $C = \{n_1, n_2, \ldots\}$ that represent the nodes of the parse tree, the log-linear model would contain, among others, the feature $f_{n_1}^{\phi_1}(\mathbf{y}) = \mathbb{I}(\vDash_{\mathbf{y}} agent(n_1) \Rightarrow left(n_1))$ that returns 1 if the contained ground formula holds in the possible world $\mathbf{y}$ and 0 otherwise.

3 MAP INFERENCE

In many settings we are given an MLN M and the state of a set of *observed* ground atoms $\left(x_{p(\mathbf{c})}\right)_{p \in O, \mathbf{c} \in C^{n_p}}$ for a set of observable predicates O. We are then asked to find the set of *hidden* ground atoms $\hat{\mathbf{y}} \in \mathcal{Y}_{H,C}$ for a set of remaining predicates $H = P \setminus O$ with maximum *a posteriori* probability (MAP)[2]

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}_{H,C}} p(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}_{H,C}} s(\mathbf{y}, \mathbf{x})$$

where

$$s(\mathbf{y}, \mathbf{x}) = \sum_{(\phi, w) \in M} w \sum_{\mathbf{c} \in C^{n_\phi}} f_{\mathbf{c}}^{\phi}(\mathbf{y}, \mathbf{x}) \qquad (2)$$

can be considered as a linear *discriminant* or *scoring* function that evaluates the goodness of a problem solution pair $(\mathbf{x}, \mathbf{y})$.

For example, we might be looking for the truth states of the $H = \{agent\}$ atoms while knowing the state of all $O = \{left\}$ ground atoms that indicate which constituents are placed to the left of the verb.

3.1 MAXWALKSAT

Previous work [Richardson and Domingos, 2005] finds $\hat{\mathbf{y}}$ using MaxWalkSAT (MWS), an approximate Random Walk method that has been very successfully used to solve Weighted SAT Problems [Kautz et al., 1996].

It starts by assigning a random state to all ground atoms and proceeds by repeatedly picking a random unsatisfied ground clause. With probability q a random ground atom of this clause is picked and its state is flipped. With probability $1 - q$ the ground atom which, when flipped, causes the largest increase in total weight $s(\mathbf{y}, \mathbf{x})$ is chosen to be flipped. The process is repeated until a fixed number n_{flips} of flips is reached. Optionally one can try again $n_{restarts}$ times to find a better $\mathbf{y}$, each time starting at a new random solution.

[2]In the case where multiple maxima exist we can pick any of these.

3.2 INTEGER LINEAR PROGRAMMING

Integer Linear Programming [ILP, Winston and Venkataramanan, 2003] refers to the process of optimising a linear objective function under a set of linear inequalities and the constraint that all (or some) variables are integers. ILP has been used in many tasks to solve MAP problems [Roth and Yih, 2005, Clarke and Lapata, 2007] because of its exactness, its declarative nature and the availability of very efficient ILP solvers.

Here we present a generic mapping from Ground Markov Networks in Markov Logic to ILPs.[3] We start by replacing each feature function application $f_{\mathbf{c}}^{\phi}(\mathbf{y})$ in equation 2 with a binary variable $\lambda_{\mathbf{c}}^{\phi}$ and constrain $f_{\mathbf{c}}^{\phi}(\mathbf{y})$ and $\lambda_{\mathbf{c}}^{\phi}$ to be equal. This leads to the optimisation problem

$$\begin{aligned} \arg\max_{\mathbf{y} \in \mathcal{Y}_{\mathbf{H},\mathbf{C}}} \quad & \sum_{(\phi, w) \in M} w \sum_{\mathbf{c} \in C^{n_\phi}} \lambda_{\mathbf{c}}^{\phi} \\ s.t \quad & \lambda_{\mathbf{c}}^{\phi} = f_{\mathbf{c}}^{\phi}(\mathbf{y}, \mathbf{x}) \, \forall (\phi, w) \in M, \mathbf{c} \in C^{n_\phi} \end{aligned}$$

with a linear objective function under a set of constraints.

In order to turn this into an ILP we need to transform each constraint into a set of linear constraints over $\mathbf{y}$ and the auxiliary variables $\left(\lambda_{\mathbf{c}}^{\phi}\right)_{\phi, \mathbf{c} \in C^{n_\phi}}$. This can be achieved by

1. Mapping each constraint to a logical equivalence of auxiliary variable and ground formula, such as $\lambda_{n_1}^{\phi} \Leftrightarrow agent(n_1) \Rightarrow left(n_1)$

2. Replacing ground atoms by either *true* or *false* if they are observed or, if not, by their corresponding decision variable, as in $\lambda_{n_1}^{\phi} \Leftrightarrow y_{n_1}^{agent} \Rightarrow false$

3. Transforming the logical equivalence into Conjunctive Normal Form[4], as in $\left(\neg\lambda_{n_1}^{\phi} \vee y_{n_1}^{agent}\right) \wedge \left(\lambda_{n_1}^{\phi} \vee \neg y_{n_1}^{agent}\right)$

4. Replacing each disjunction by a linear constraint [Williams, 1999], for example $-1.0 \cdot \lambda_{n_1}^{\phi} + 1.0 \cdot y_{n_1}^{agent} \geq 0$

Note that we can significantly simplify the above program for hard constraints (i.e.., formulae with very large w) and formulae with only one hidden atom. We omit details for brevity.

[3]In general it is always possible to map any Markov Network to an ILP [Taskar, 2004]. However, our mapping is tailor-made for Markov Logic and yields smaller programs.

[4]If the formula contains universal (existential) quantified formulae we replace these with conjunctions (disjunctions) using the finite set of constants C.

4 CUTTING PLANE INFERENCE

We will show in the empirical section of this paper that running inference using the full grounding of a Markov Logic Network can be slow and in the case of MWS also inaccurate. We will now present an algorithm that tries to overcome this problem by instantiating only portions of the complete Ground Markov Network and running an off-the-shelf inference method in this network.

4.1 ALGORITHM

The proposed algorithm is a variant of the Cutting Plane approach from Operations Research [Dantzig et al., 1954]. Cutting Plane algorithms solve large scale constrained optimisation problems by only considering a subset of constraints. In each iteration the solution to a partial problem is provided to an oracle that returns a set of constraints[5] the solution violates. The current problem is extended by these new constraints and re-solved. The process is repeated until no more violated constraints can be found.

Instead of searching for violated constraints, *Cutting Plane Inference* (CPI) searches for feature-weight products in equation 2 that do not maximally contribute to the overall sum given the current solution. More precisely, for each formula ϕ and a given $(\mathbf{y}', \mathbf{x})$ we are looking for all tuples, $\text{Separate}\,(\phi, w, \mathbf{y}, \mathbf{x}) \subseteq C^{n_\phi}$, for which

$$w \cdot f_{\mathbf{c}}^{\phi}\,(\mathbf{y}', \mathbf{x}) < \max_{\mathbf{y} \in Y_{H,C}} w \cdot f_{\mathbf{c}}^{\phi}\,(\mathbf{y}, \mathbf{x}) \tag{3}$$

We will say that the corresponding ground formulae are *not maximally satisfied* in the world $\mathbf{y}'$.

In the terminology of the Cutting Plane method this step is often referred to as *separation*: it finds a set of constraints that separates feasible solution from infeasible solutions. In our case this step will help to separate possible worlds with high score from those with low score.

It will be useful to define a *partial grounding* $\mathbf{G} = (G_\phi)_{(\phi,w)\in M}$ with $G_\phi \subseteq C^{n_\phi}$ that maps each first order formula ϕ to a set of tuples we ground it with. A partial grounding induces a *partial score*

$$s_{\mathbf{G}}\,(\mathbf{y}, \mathbf{x}) = \sum_{(\phi,w)\in M} w \sum_{\mathbf{c}\in G_\phi} f_{\mathbf{c}}^{\phi}\,(\mathbf{y}, \mathbf{x}) \tag{4}$$

CPI proceeds as described in algorithm 1. In each iteration i we maintain a partial grounding $\mathbf{G}^i$. Initially $\mathbf{G}^0$ is filled with a small number of groundings. A natural choice are all groundings of formulae that only contain one hidden predicate. In this case maximising $s_{\mathbf{G}^0}$ is trivial because the hidden variables do not interact and often gives a very good first guess.

In step 5 we find a solution $\mathbf{y}$ that maximises the partial score $s_{\mathbf{G}^{i-1}}$ (or approximately maximises it). For this we can pick our optimisation method of choice. In steps 9 and 10 we find the ground formulae which are not maximally satisfied in the current solution $\mathbf{y}$ and add them to the current partial grounding. We terminate if no more new ground formulae are foundor a maximum number of iterations is reached. This process calculates one solution $\mathbf{y}$ in each iteration. The final result is the solution $\mathbf{y}$ with highest score.

Algorithm 1 $\text{CPI}\big(M, \mathbf{G}^0, \mathbf{x}\big)$

```
1: i ← 0
2: y' ← 0
3: repeat
4:    i ← i + 1
5:    y ← solve(G^{i-1}, x)
6:    if s(y, x) > s(y', x) then
7:       y' ← y
8:    end if
9:    for each (φ, w) ∈ M do
10:      G_φ^i ← G_φ^{i-1} ∪ Separate(φ, w, y, x)
11:   end for
12: until G_φ^i = G_φ^{i-1} or i > maxIterations
13: return y'
```

The following theorem shows that when CPI returns the solution of iteration i the error is bound by the sum of the error of the base solver on the partial problem and the sum of absolute weights of newly found ground formulae at this iteration. In particular, for an iteration with no more newly found groundings the error is only bound by (in fact it is equal to) the error of the base solver on the partial problem, which is likely to be much smaller and easier to solver than the original one.

This also shows that if the base solver is exact (like ILP) and no more groundings are found, CPI will be exact. If we choose a solution for which new ground formulae were found the error bound is incremented by the sum of the absolute weights of these ground clauses. Thus we still do well if the remaining clauses have small weight.

Theorem. *Let* $\hat{\mathbf{y}}$ *be an optimal solution,* $\mathbf{y}'$ *the solution returned by CPI taken from iteration* i*,* $\hat{\mathbf{y}}_{\mathbf{G}^{i-1}}$ *an optimal solution for* $s_{\mathbf{G}^{i-1}}$ *and* $b = \sum_{(\phi,w)} \sum_{\mathbf{c}\in\mathbf{G}_\phi^i \setminus \mathbf{G}_\phi^{i-1}} |w|$ *then*

$$\begin{aligned} & s\,(\hat{\mathbf{y}}, \mathbf{x}) - s\,(\mathbf{y}', \mathbf{x}) \\ \leq\ & s_{\mathbf{G}^{i-1}}\,(\hat{\mathbf{y}}_{\mathbf{G}^{i-1}}, \mathbf{x}) - s_{\mathbf{G}^{i-1}}\,(\mathbf{y}', \mathbf{x}) + b \end{aligned}$$

Proof. Let $\mathbf{G}^i \setminus \mathbf{G}^{i-1} = \left(G_\phi^i \setminus G_\phi^{i-1}\right)_\phi$ be the newly

[5] In case of linear constraints these constraints form hyperplanes that further *cut* the space of feasible solution, hence the name.

added groundings and $\overline{\mathbf{G}^i} = \left(C^{n_\phi} \setminus G^i_\phi\right)_\phi$ the remaining groundings. We can split $s(\hat{\mathbf{y}}, \mathbf{x}) - s(\mathbf{y}', \mathbf{x})$ into three parts, a score difference for the ground formulae in $\mathbf{G}^{i-1}$, those in $\mathbf{G}^i \setminus \mathbf{G}^{i-1}$ and $\overline{\mathbf{G}^i}$. We know that $\mathbf{y}'$ solves $s_{\overline{\mathbf{G}^i}}$ optimally based on equation 3, thus $s_{\overline{\mathbf{G}^i}}(\hat{\mathbf{y}}, \mathbf{x}) - s_{\overline{\mathbf{G}^i}}(\mathbf{y}', \mathbf{x}) \leq 0$. Furthermore, in the worst case each term $w \cdot f^\phi_{\mathbf{c}}(\mathbf{y}', \mathbf{x})$ in $s_{\mathbf{G}^i/\mathbf{G}^{i-1}}(\mathbf{y}', \mathbf{x})$ is by $|w|$ smaller than each corresponding term in $s_{\mathbf{G}^i \setminus \mathbf{G}^{i-1}}(\hat{\mathbf{y}}, \mathbf{x})$, leading to $s_{\mathbf{G}^i \setminus \mathbf{G}^{i-1}}(\hat{\mathbf{y}}, \mathbf{x}) - s_{\mathbf{G}^i \setminus \mathbf{G}^{i-1}}(\mathbf{y}', \mathbf{x}) \leq \sum_{(\phi,w)} \sum_{\mathbf{c} \in \mathbf{G}^i_\phi \setminus \mathbf{G}^{i-1}_\phi} |w|$. □

Note that we do not make any claims about the runtime of the algorithm. Even without a limit on the number iterations it is guaranteed to converge in a finite number of steps due to the fact that the solution space is finite and we will either try each solution or return to a previous one. However, we cannot provide guarantees as to how many steps this will take. Thus we allow the algorithm to terminate before convergence is reached.

4.2 SEPARATION

An integral part of CPI is the separation step, in which we need to find all groundings $\mathbf{c}$ of a formula ϕ and weight w which are not maximally satisfied (according to equation 3) for a given solution $\mathbf{y}'$. It is this step for which the Statistical Relational Learning paradigm comes into play. In a (propositional) Markov Network we do not have any higher order descriptions of its features. Performing separation then means evaluating all features of the network.

In Markov Logic, however, we can do better. There are two cases to consider. If $w > 0$ we have to find assignments $\mathbf{c}$ with $f^\phi_{\mathbf{c}}(\mathbf{y}, \mathbf{x}) = 0$, that is, groundings for which $\vDash_{\mathbf{y},\mathbf{x}} \phi[\mathbf{v}/\mathbf{c}]$ is false. Correspondingly, for $w < 0$ we have to find $\mathbf{c}$ for which $\vDash_{\mathbf{y},\mathbf{x}} \phi[\mathbf{v}/\mathbf{c}]$ is true.

We cast this into a database query evaluation problem[6] and store the atoms in $\mathbf{y}$ and $\mathbf{x}$ as rows in database tables. Then we convert the formula ϕ (or $\neg\phi$) to a database query which is executed during CPI. Such queries can often be processed very efficiently [Grohe et al., 2001]. In our experiments the cost of query evaluation was marginal when compared to the cost of numeric optimisation.

4.3 RELATED WORK

The idea of Cutting Planes have been used in at least two ways. We can either use it to tackle ILP problems by solving their LP relaxation and, in case the solution is fractional, generate additional constraints the integer solution is known to fulfil. Or we use it to solve problems with a large number of constraints, such as ILP formulations of the Travelling Salesman Problem [Dantzig et al., 1954], without having to include all of them.

Our work follows previous research in MAP inference [Riedel and Clarke, 2006, Anguelov et al., 2004, Tromble and Eisner, 2006, Sontag and Jaakkola, 2007] that uses Cutting Planes to avoid including all constraints in advance. However, in this work we frame, implement and evaluate the approach more generally as a meta algorithm for MAP inference in Markov Logic Networks into which we can plug-in any existing propositional solver. This includes the introduction of a separation routine that does not require additional implementation efforts when applied to new tasks. Markov Logic Networks may also contain nondeterministic constraints. In contrast to previous work [Tromble and Eisner, 2006] our method handles these without the need to branch-and-bound.

CPI is also similar in nature to LazySAT [Singla and Domingos, 2006b], a memory-efficient implementation of MWS: both methods avoid to instantiate the full ground network. However, while CPI only instantiates new parts of the ground network once the base solver has optimised the current partial network, LazySAT instantiates new parts of the network whenever they may be needed during the inner loop of MWS. Note that although CPI also reduces memory overhead, in this work we focus on its speed and accuracy and thus do not directly compare it to LazySAT, which inherits the speed and accuracy of MWS.

5 EXPERIMENTS

We use two tasks to evaluate the utility of CPI as a meta MAP inference method for Markov Logic. The first is Semantic Role Labelling, the second Joint Entity Resolution. In both cases we want to investigate how CPI affects the runtime and accuracy of two base solvers: MWS and ILP. For all experiments we use our own Markov Logic implementation running on a Pentium 4 at 2.8Ghz with 4Gb RAM. All CPI systems use local formulae with only one hidden atom to create the initial grounding $\mathbf{G}^0$.

5.1 SEMANTIC ROLE LABELLING

Semantic Role Labelling refers to the task of identifying and classifying the arguments and modifiers of verbs in natural language text, as in

[A0 He] [AM-MOD would] [A0 n't] [V accept] [A1 anything of value].

for the verb "accept". Labels such as "A0" serve as placeholders for actual roles of the given verb, such

[6] Alternatively we could frame this problem as an instance of theorem proving, but all axioms are ground atoms and we are looking for *all* groundings for which the formula holds – Database technology is optimised for this setting.

as "acceptor" in the above case. The most effective approach to Semantic Role Labelling to date is based on the output of a constituent parser. Each constituent is labelled with the type of argument or modifier it represents with respect to the verb in question. We model the task using a (typed) binary *label* predicate defined over constituents and possible labels. Atoms of this predicate are hidden at test time.

We set up a knowledge base of rules that describe local features of constituents and global dependencies between labels, resembling previous work in Semantic Role Labelling [Punyakanok et al., 2005]. The rules we use are slightly more general than our examples ϕ_1 and ϕ_2 in section 2.

We learn the weights of this model using the CoNLL 2005 dataset [Carreras and Marquez, 2005] and the Online Learner MIRA [Crammer and Singer, 2003]. For inference during training we use CPI with ILP.

For testing we use the first 100 verb frames from the WSJ test set of the CoNLL 2005.[7] In table 1 we show the following metrics: 1) the score delta to the optimal solution with respect to the soft clauses, $\Delta s_{soft} = s_{\mathbf{G}_{soft}}(\hat{\mathbf{y}}, \mathbf{x}) - s_{\mathbf{G}_{soft}}(\mathbf{y}', \mathbf{x})$ where $\mathbf{G}_{soft}$ contains all ground formulae for each non-deterministic formula; 2) the number of violations of deterministic formulae; 3) the runtime taken; 4) the number of calls to the optimiser; 5) the F1 accuracy on the task. Note that the total score delta $\Delta s = s(\hat{\mathbf{y}}, \mathbf{x}) - s(\mathbf{y}', \mathbf{x})$ is always dominated by the hard constraints as they have very large weights. Thus if system A produces one less violation than system B its total score delta Δs will be smaller.

We first note that using plain MWS with 100,000 flips (M-100k) and no restarts[8] is less accurate in terms of soft model score and F1 accuracy when compared with CPI-MWS using the same number of flips (C-M-100k). It is also significantly slower and produces some hard constraint violations while CPI & MWS does not. When using ILPwe achieve perfect model score since ILP returns exact solutions. Using ILP with CPI (C-ILP) is still exact; however, CPI speeds up the solver by almost two orders of magnitude.[9]

We also ran CPI-ILP on the full test set to compare our system with the state of the art, yielding 77.1 F1 measure. When compared to the entries in the CoNLL shared task that only use the output of one parser our system would come out first [Carreras and Marquez, 2005].

	Δs_{soft}	Viol.	t(s)	It.	F1
M-100k	-0.098	0.05	70	1	0.61
C-M-10k	-0.26	0.01	0.18	3.7	0.65
C-M-100k	-0.074	0	1.2	3.5	0.69
ILP	0	0	4.6	1	0.79
C-ILP	0	0	0.065	3.2	0.79

Table 1: Semantic Role Labelling results averaged over the first 100 examples in WSJ test set; Δs_{soft} is the soft score delta wrt to the true MAP solution; $Viol$ the number of violations; $It.$ the number of optimisation calls; $t(s)$ the time spent in seconds; F1 is F1 accuracy.

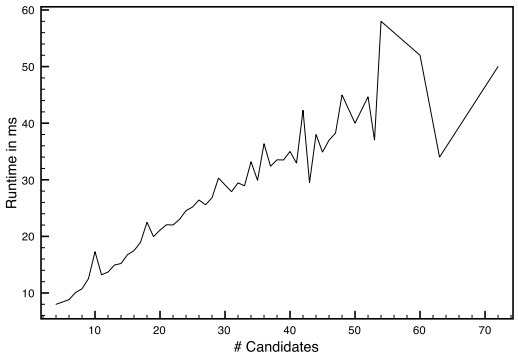


Figure 1: Runtime of CPI-ILP for Semantic Role Labelling for different numbers of candidate constituents.

We also wanted to investigate how CPI scales with problem size. Figure 1 shows the runtime of CPI-ILP against the number of candidate nodes. It seems that for this dataset and problem, CPI scales almost linearly with problem size up until 50 candidates. After 50 candidates a linear trend can only be guessed due to data sparseness. This linear dependency is interesting because the actual number of ground formulae rises quadratically (due to the no-overlap formula and others).

5.2 JOINT ENTITY RESOLUTION

Entity resolution is a crucial problem in many business, government and research projects. It can be described as the task of finding database records that refer to the same entity and is very similar to Coreference Resolution in NLP. In our experiments records are citations, and we search for citations describing the same publication; however, we not only want to find matching citations, we also want to jointly identify author or venue name strings referring to the same author or venue, respectively.

We use a knowledge base with 46 formulae provided

[7] The reason for not using more instances were memory problems we encountered when we were using MWS alone and grounding the complete network. These problems will likely disappear when using LazySAT instead of MWS.

[8] Note that we also experimented with using restarts; however, differences to runs with equivalent total numbers of flips and no restarts were only marginal.

[9] Note that the difference in runtime between MWS and ILP should be taken with caution: for ILP we use a well-established software library (lp-solve), for MWS our own implementation.

	Δs_{soft}	Viol.	t(m)	It.	F1
M-100k	-4578	704.5	2.67	1	0.30
C-M-1k	-2446	796.1	0.55	30	0.69
C-M-10k	-2682	594.9	0.96	30	0.70
C-ILP	0	0	1.56	5.9	0.72

Table 2: Averaged results over 10 folds of the Cora dataset;Δs_{soft} is the soft score delta wrt to the true MAP solution; $Viol$ the number of violations; $It.$ the number of optimisation calls; t(m) the time spent in minutes; F1 is F1 accuracy.

by Singla and Domingos [2005] with predicates such as *sameBib* and *sameAuthor* that denote citation and author matches, respectively. The knowledge base states regularities such as "if two authors names match the corresponding citations match" or "if the tdf-if distance between the titles is between 0.7 and 0.8 the titles match". It also contains the transitivity rule

$$sameBib(v_1, v_2) \wedge sameBib(v_2, v_3) \Rightarrow sameBib(v_1, v_3)$$

This is a hard constraint and imposes a difficult problem for many generic inference methods [Poon and Domingos, 2006].

In our first set of experiments we used a cleaned version [Singla and Domingos, 2005] of the Cora Database [Bilenko and Mooney, 2003], containing approximately 1200 citations of computer science articles. In total these citations refer to about 120 unique publications. Following previous work [Singla and Domingos, 2006b], we tested and trained using a 10-fold leave-one-out procedure and Pseudo-Likelihood estimation while ensuring that folds do not contain split citation clusters [Singla and Domingos, 2005]. Each fold contains roughly 120 records.

Table 2 shows our results for Entity Resolution. They are consistent with those in table 1: again CPI renders MWS faster and more accurate both in terms of violations, soft model score and F1 measure. However, this time we cannot directly compare plain ILP with CPI-ILP because the full ILP did not fit into memory. In other words, here CPI makes an infeasible method feasible. Note that in this case CPI-MWS did not converge, thus we terminated the algorithm after a predefined number of iterations (30).

Interestingly, the F1 accuracy of CPI with MWS is significantly better than the F1 accuracy of MWS alone. This can be explained if we consider that CPI-MWS returns solutions with significantly higher soft model score. This score reflects what the model learnt to be a good matching, independent of the number of violations. CPI-MWS can achieve a higher soft score because it starts at a solution that maximises the local score without considering any hard constraints. MWS, on the other hand, starts at a random solution

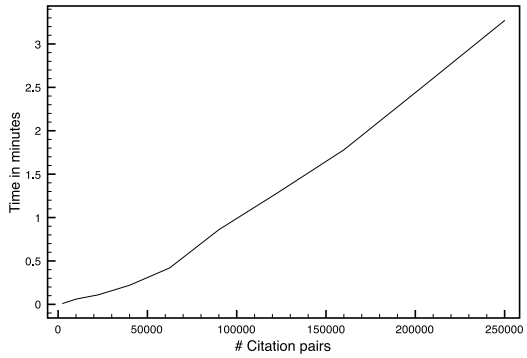


Figure 2: Runtime of CPI-ILP for different subsets of Bibserv.org, averaged over 5 instances for each number of citation pairs.

that may contain many (locally) unlikely matchings and will spend most of its time making these unlikely matchings consistent.

In our second set of experiments we wanted to again evaluate the runtime behaviour of CPI when the problem size increases. For this purpose we used the Bibserv.org corpus and a model trained on the Cora dataset. Bibserv.org consists of about 20,000 citations. We used the same random subsets of size 50 to 500 in steps of 50 as Singla and Domingos [2006b]. All following results are averaged over 5 datasets of the same size.

Figure 2 shows the runtime of CPI-ILP with increasing number of citation pairs – this corresponds to the number of decisions to make. Again CPI-ILP seems to scale linearly with the number of variables and thus quadratically with the number of citations. Yet, the number of features in the complete network scales at least cubically with the number of citations due to the transitivity clause.

6 CONCLUSION

In this paper we presented Cutting Plane Inference (CPI), a novel method for finding MAP solutions in Markov Logic that incrementally solves partial versions of the complete Ground Markov Network based on a Cutting Plane approach. While Cutting Planes have been used for specific MAP inference problems before, this work shows how they can be generalised and incorporated into a Statistical Relational Learning framework where they become automatically available for a wide range of tasks. Our method essentially serves as a *meta* algorithm that alternates between deterministic first order query processing on one hand, and numeric optimisation of partial problems on the other.

We evaluated the proposed algorithm using two real-

world tasks for which we showed MWS to perform poorly: Joint Entity Resolution and Semantic Role Labelling. In both cases CPI makes an exact method (ILP) more efficient while remaining exact, and an approximate method (MWS) both faster and more accurate.

However, exact MAP inference in general Graphical Models is PP-complete [Park, 2002]. Thus we obviously cannot expect Cutting Plane Inference to work for arbitrary formulae, weights and problems – at least not for ILP as base solver. Yet, we believe that both of the above tasks are instances of a larger class of problems that are not so much characterised by their network structure or strength of weights but by how well local formulae and weights predict the global goodness of a structure. CPI extends the applicability of SRL to this class and might therefore contribute to a more widespread use of SRL.

It will be important to investigate how to characterise the class of problems we can not solve with CPI. For example, consider a conjunctive formula like $p(v_1) \wedge q(v_2)$ with positive weight and sparsely populated predicates p and q in the current solution $\mathbf{y}$. Separation will find all pairs of v_1 and v_2 for which the conjunction does not hold and here this set would be almost exhaustive, resulting in a problem not much smaller than the original one.

References

D. Anguelov, D. Koller, P. Srinivasan, S. Thrun, H.-C. Pang, and J. Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *Advances in Neural Information Processing Systems*, 2004.

M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proc. of the Intl. Conf. on Knowledge Discovery and Data Mining*, 2003.

X. Carreras and L. Marquez. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proc. of the Conf. on Computational Natural Language Learning*, 2005.

James Clarke and Mirella Lapata. Modelling compression with discourse constraints. In *Proc. of the 2007 Joint EMNLP/CoNLL Conf.*, 2007.

Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 2003.

G. B. Dantzig, R. Fulkerson, and S. M. Johnson. Solution of a large-scale traveling salesman problem. *Operations Research*, 1954.

R. de Salvo Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In *Proc. of the 2005 Intl. Joint Conf. on Artificial Intelligence*, 2005.

P. Domingos. *Artificial Intelligence: The. First Hundred Years*, chapter What's Missing in AI: The Interface Layer. AAAI Press, 2006.

Pedro Domingos. Structured machine learning: Ten problems for the next ten years. In *Proc. of the Annual Intl. Conf. on Inductive Logic Programming*, 2007.

Michael Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, 1987.

Martin Grohe, Thomas Schwentick, and Luc Segoufin. When is the evaluation of conjunctive queries tractable? In *Proc. of the ACM Symposium on Theory of Computing*, 2001.

H. Kautz, B. Selman, and Y. Jiang. A general stochastic approach to solving problems with hard and soft constraints. In *The Satisfiability Problem: Theory and Applications*, 1996.

D. Koller. Probabilistic relational models. In *Proc. of the Intl. Conf. on Inductive Logic Programming*, 1999.

Raymond T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 1992.

J. Park. Map complexity results and approximation methods. In *Proc. of the Conf. on Uncertainty in AI*, 2002.

H. Poon and P. Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proc. of the Natl. Conf. on Artificial Intelligence*. AAAI Press, 2006.

Hoifung Poon and Pedro Domingos. Joint inference in information extraction. In *Proc. of the Natl. Conf. on Artificial Intelligence*, 2007.

V. Punyakanok, D. Roth, and W. Yih. Generalized inference with multiple semantic role labeling systems. In *Proc. of the Conf. on Computational Natural Language Learning*, 2005.

Matthew Richardson and Pedro Domingos. Markov logic networks. Technical report, University of Washington, 2005.

Sebastian Riedel and James Clarke. Incremental integer linear programming for non-projective dependency parsing. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, 2006.

D. Roth and W. Yih. Integer linear programming inference for conditional random fields. In *Proc. of the Intl. Conf. on Machine Learning*, 2005.

Parag Singla and Pedro Domingos. Discriminative training of markov logic networks. In *Proc. of the Natl. Conf. on Artificial Intelligence*, 2005.

Parag Singla and Pedro Domingos. Entity resolution with markov logic,. In *Proc. of the Intl. Conf. on Data Mining*, 2006a.

Parag Singla and Pedro Domingos. Memory-efficient inference in relational domains. In *Proc. of the Natl. Conf. on Artificial Intelligence*, 2006b.

D. Sontag and T. Jaakkola. New outer bounds on the marginal polytope. In *Advances in Neural Information Processing Systems*, 2007.

Ben Taskar. *Learning Structured Prediction Models: a Large-Margin approach*. PhD thesis, Stanford University, 2004.

Roy W. Tromble and Jason Eisner. A fast finite-state relaxation method for enforcing global constraints on sequence decoding. In *Proc. of the Joint HLT/NAACL Conf.*, 2006.

H. Paul Williams. *Model Building in Mathematical Programming*. Wiley, 4th edition, 1999.

Wayne L. Winston and Munirpallam Venkataramanan. *Introduction to Mathematical Programming*. Brooks/Cole, 2003.

Model-Based Bayesian Reinforcement Learning in Large Structured Domains

Stéphane Ross
School of Computer Science
McGill University
Montreal, Canada

Joelle Pineau
School of Computer Science
McGill University
Montreal, Canada

Abstract

Model-based Bayesian reinforcement learning has generated significant interest in the AI community as it provides an elegant solution to the optimal exploration-exploitation tradeoff in classical reinforcement learning. Unfortunately, the applicability of this type of approach has been limited to small domains due to the high complexity of reasoning about the joint posterior over model parameters. In this paper, we consider the use of factored representations combined with online planning techniques, to improve scalability of these methods. The main contribution of this paper is a Bayesian framework for learning the structure and parameters of a dynamical system, while also simultaneously planning a (near-)optimal sequence of actions.

1 Introduction

In the past decades, reinforcement learning (RL) has emerged as a useful technique for learning how to optimally control systems with unknown dynamics (Sutton & Barto, 1998). However classical RL has many shortcomings. In particular, RL does not address the problem of how to efficiently gather data to learn the parameters of the system, as well as how to behave in systems where the costs incurred during learning matter, i.e. the well known exploration-exploitation tradeoff problem. These shortcomings are mostly related to the fact that classical RL does not consider the uncertainty in the learned parameters for decision-making, nor does it allow for flexibly including prior knowledge about the system's dynamics.

Model-based Bayesian RL methods have successfully addressed these issues by maintaining a posterior distribution over unknown model parameters and acting such as to maximize long-term expected rewards with respect to this posterior (Dearden, Friedman, & Andre, 1999; Duff, 2002; Poupart, Vlassis, Hoey, & Regan, 2006). Prior knowledge of the system can be defined explicitly by specifying a prior distribution over model parameters. This allows for a flexible way of encoding uncertain knowledge into the learning algorithm. Furthermore, if the resulting decision problem is solved exactly, this provides an optimal exploration-exploitation tradeoff, in that the agent will behave such as to maximize long-term expected rewards with respect to the prior.

However, due to the high complexity of model-based Bayesian RL, most approaches have been limited to very small domains (10-20 states). This is mainly due to two reasons. First, when the number of states is large, a large amount of data needs to be collected to learn a good model, unless very few parameters are unknown or some structural assumptions are made to represent the dynamics with few parameters. Second, most planning approaches in Bayesian RL become intractable as the number of states increases, since planning is done over the full space of possible posteriors.

To address the first issue, we propose learning a factored representation of the dynamics via a Bayesian approach. Factored representations can efficiently represent the dynamics of a system with fewer parameters using a dynamic Bayesian network (DBN) that exploits conditional independence relations existing between state features (Boutilier, Dearden, & Goldszmidt, 2000; Guestrin, Koller, Parr, & Venkataraman, 2003). Bayesian RL techniques can be extended quite easily to factored representations when the structure of this DBN is known, however this is unreasonable in many domains. Fortunately, the problem of simultaneously learning the structure and parameters of a Bayes Net has received some attention (Heckerman, Geiger, & Chickering, 1995; Friedman & Koller, 2003; Eaton & Murphy, 2007), which we can leverage for our work. However while these approaches provide an effective

way of learning the model, it is far from sufficient for Bayesian RL, where the goal is to *choose actions* in an optimal way, with respect to what we have learned about the model.

To address the issue of action selection, we propose incorporating an online Monte Carlo approach to evaluate sequences of actions with respect to the posterior over structures and parameters. The focus on online (rather than offline) planning means that we only need to plan with respect to the current posterior (rather than all possible posteriors), which offers substantial computational savings.

The main contribution of this paper is a novel Bayesian framework for optimizing the choice of actions in a structured dynamical system, with unknown structure and parameters. We present experimental results of our approach in a variety of large network administration domains, showing good performance for problems with thousands of states.

2 Background

A Markov Decision Process (MDP) is a general framework for decision making in stochastic systems (Bellman, 1957). It is often used to represent reinforcement learning problems (Sutton & Barto, 1998). We consider an unknown system represented by an MDP model in factored form (S, A, T, R) where:

- $S : S_1 \times S_2 \times \cdots \times S_n$, is the (discrete) set of states of the system; $S_1, \ldots, S_n$ correspond to the domain of the n state variables (features).
- A, the (discrete) set of actions that can be performed by the agent.
- $T : S \times A \times S \to [0, 1]$, the transition function, where $T(s, a, s') = \Pr(s'|s, a)$ represents the probability of moving to state s' if the agent executes action a in state s. This can be represented efficiently by a DBN for each action, exploiting conditional independence relations that exist between state features (Boutilier et al., 2000). For simplicity, we assume that these DBNs are bipartite graphs, so dependencies only exist between state variables at time t and state variables at time $t+1$.
- $R : S \times A \to \mathbb{R}$, the reward function, defined for every action of the agent in every state.

The DBN defining T for any action $a \in A$ is represented by a graph G_a and set of parameters θ_{G_a} defining the conditional probability tables. For any state variable s'_i, we denote its set of parent variables in this graph by $\mathrm{Par}_i(G_a)$ and given the previous state s, the values of these parents variables by $\mathrm{ParVal}_i(s, G_a)$. For each possible value $v \in S_i$ of state variable s'_i, and each possible assignment to its parent values $E \in S_{\mathrm{Par}_i(G_a)} = \prod_{j \in \mathrm{Par}_i(G_a)} S_j$, θ_{G_a} contains a parameter $\theta_{G_a}^{i,v|E}$ that defines $\Pr(s'_i = v|\mathrm{ParVal}_i(s, G_a) = E, a)$. Given such graph G_a and parameters θ_{G_a}, $T(s, a, s')$ is computed efficiently as:

$$T(s, a, s') = \prod_{i=1}^{n} \Pr(s'_i|\mathrm{ParVal}_i(s, G_a), a). \quad (1)$$

The goal of the MDP agent is to find an action selection strategy, called a *policy*, that maximizes its long-term expected rewards. The optimal action to take in a state s is defined via the optimal value function V^* representing the return obtained by the optimal policy starting in state s:

$$V^*(s) = \max_{a \in A} \left[R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s') \right]. \quad (2)$$

The optimal action in s is obtained by taking the arg max instead of the max in the last equation. In general, a factored representation of the transition does not induce a structured representation of the optimal value function. However, approximate algorithms exist to compute V^* more efficiently by exploiting the factored representation (Guestrin et al., 2003).

2.1 Bayesian Reinforcement Learning

While the MDP framework allows one to compute the optimal policy for any stochastic system, it requires full knowledge of the transition dynamics. This is a strong assumption in practice. Model-based Bayesian RL weakens this assumption by instead maintaining a probability distribution over the possible settings of each unknown parameter (Dearden et al., 1999). It assumes an initial prior distribution over these parameters, and uses Bayes' rule to update the posterior distribution whenever state-transitions are observed in the course of interactions between the agent and the environment. Given that transition parameters are usually modeled using multinomial distributions, a natural choice to specify this posterior is the Dirichlet distribution. The Dirichlet is specified by "count" parameters, $\phi_1, \ldots, \phi_m$, specifying the likelihood $f(p|\phi)$ that outcomes $1, \ldots, m$ occur with probabilities $p_1, \ldots, p_m$ given they were observed $\phi_1, \ldots, \phi_m$ times:

$$f(p|\phi) = \frac{1}{B(\phi)} \prod_{i=1}^{m} p_i^{\phi_i - 1}, \quad (3)$$

where $B(\phi)$ is the multinomial beta function. This choice of prior allows for a flexible way to input prior knowledge in the system, as well as an easy way to maintain the posterior. We refer to the set of counts ϕ

for all possible transitions (s, a, s') as the information state of the agent. The resulting decision problem is the following: given the agent is in state s with information state ϕ, how should it behave such as to maximize its future expected rewards? This new decision problem can be modeled by an extended MDP model, called Bayes-Adaptive MDP (BAMDP), where the counts ϕ are included in the state space, and the transition function models how these parameters evolve given a particular state transition (Duff, 2002). This extended MDP has infinitely many states but can be solved exactly over a finite horizon for any particular current state and information state.

2.2 Learning Bayes Nets

Bayesian networks (BNs) have been used extensively to build compact predictive models of multivariate data. A BN models the joint distribution of multivariate data compactly by exploiting conditional independence relations between variables. It is defined by a set of variables X, a directed acyclic graph (DAG) structure G over variables in X, and parameters θ_G, where $\theta_G^{i,v|E}$ specifies the probability that $X_i = v$ given that its parents in G take value E.

Several approaches exist to learn BNs. Learning a Bayes net can involve either only learning θ_G (if the structure G is known), or simultaneously learning the structure G and parameters θ_G. For our purposes, we are mostly interested in Bayesian approaches that learn both the structure and parameters (Heckerman et al., 1995; Friedman & Koller, 2003; Eaton & Murphy, 2007). These Bayesian approaches proceed by first specifying a joint prior, $P(G, \theta_G)$, of the form:

$$P(G, \theta_G) = P(G)P(\theta_G|G), \tag{4}$$

where $P(G)$ is a prior over structures and $P(\theta_G|G)$ is a conditional prior on the parameters θ_G given a particular structure G. $P(G)$ is often chosen to be uniform, or proportional to $\beta^{|E(G)|}$ for some $\beta \in (0, 1)$ where $|E(G)|$ is the number of edges in G, such as to favor simpler structures.

It follows that if dataset D is observed, then the joint posterior is defined as follows:

$$P(G, \theta_G|D) = P(G|D)P(\theta_G|G, D). \tag{5}$$

To compute this posterior efficiently, several assumptions are usually made about the prior $P(\theta_G|G)$. First, it should factorize into a product of independent Dirichlet priors:

$$\begin{aligned} P(\theta_G|G) &= \prod_{i=1}^{n} \prod_{E \in S_{\mathrm{Par}_i(G)}} P(\theta_G^{i,*|E}|G), \\ P(\theta_G^{i,*|E}|G) &\sim Dirichlet(\phi_G^{i,*|E}), \end{aligned} \tag{6}$$

Under this independence assumption, the term $P(\theta_G|G, D)$ is a product of Dirichlet distributions, which can be updated easily by incrementing counts $\phi_G^{i,v|E}$ for each $X_i = v|\mathrm{ParVal}_i(G) = E$ in D.

A second common assumption is that two equivalent graph structures G and G' should have equivalent priors over θ_G and θ'_G (this is called the *likelihood equivalence assumption*). This enforces a strong relation between the priors $P(\theta_G|G)$ and $P(\theta_{G_c}|G_c)$ for the complete graph G_c (where every variable depends on all previous variables). Hence specifying ϕ_{G_c} totally specifies the prior on θ_G for any other graph G.

For many problems, the posterior $P(G|D)$ cannot be maintained in closed form as it corresponds to a discrete distribution over $O(n!2^{\binom{n}{2}})$ possible graph structures. Instead, MCMC algorithms can be used to sample graph structures from this posterior (Friedman & Koller, 2003). The well known Metropolis-Hasting algorithm specifies that a move from graph G to G' should be accepted with probability $\min\left\{1, \frac{P(D|G')P(G')q(G|G')}{P(D|G)P(G)q(G'|G)}\right\}$, where $q(G'|G)$ is the probability that a move from G to G' is proposed and $P(D|G) = \int P(D|G, \theta_G)P(\theta_G|G)d\theta_G$. Such random walk in the space of DAGs has the desired stationary distribution $P(G|D)$. Under previous assumptions concerning the prior $P(\theta_G|G)$, $P(D|G)$ can be computed in closed form and corresponds to the likelihood-equivalence Bayesian Dirichlet score metric (BDe) (Heckerman et al., 1995). Typical moves considered include adding an edge, deleting an edge, or reversing an edge in G.

3 Bayesian RL in Factored MDPs

We consider the problem of acting optimally in a system represented as a factored MDP, in the case where both the structure and parameters of the DBNs defining the transition function, T, are unknown. We assume that the state features $S_1, \ldots, S_n$, the action set A, and the reward function R, are known. Our work extends trivially to the case where R is unknown, but we leave this out for simplicity of presentation.

3.1 Factored Bayesian RL model

We consider the transition function T as a hidden variable of the system, which is partially observed whenever state transitions occur in the system. In this view, the decision problem can be cast as a Partially Observable MDP (POMDP) (Kaelbling, Littman, & Cassandra, 1998). The state of this POMDP captures both the actual system state, and the DBNs defining T for each action $a \in A$. Formally, this POMDP is defined

by the tuple (S', A', Z', T', O', R'):

- $S' : S \times \mathcal{G}^{|A|}$, where S is the original state space of the MDP, $\mathcal{G}$ is the set of DBNs (G, θ_G) (one per action) and G is a bipartite graph from $S_1, \ldots, S_n$ to $S_1, \ldots, S_n$.
- $A' = A$, the set of actions in the original MDP.
- $Z' = S$, the set of observations (i.e a transition to a particular state of the MDP)
- $T' : S' \times A' \times S' \to [0, 1]$, the transition function in this POMDP, where:

$$\begin{aligned} & T'(s, G, \theta_G, a, s', G', \theta'_{G'}) \qquad (7) \\ &= \Pr(s', G', \theta'_{G'} | s, G, \theta_G, a) \\ &= \Pr(s' | s, G, \theta_G, a) \Pr(G', \theta'_{G'} | G, \theta_G, s, a, s'). \end{aligned}$$

  Since we assume that the transition function does not change over time, then

$$\Pr(G', \theta'_{G'} | G, \theta_G s, a, s') = I_{(G,\theta_G)}(G', \theta'_{G'})$$

  (the indicator function of (G, θ_G)), and

$$\Pr(s' | s, G, \theta_G, a) = \prod_{i=1}^{n} \theta_{G_a}^{i, s'_i | \mathrm{ParVal}_i(s, G_a)}.$$

- $O' : S' \times A' \times Z' \to [0, 1]$, the observation function, where $O(s', G', \theta'_{G'}, a, z)$ is the probability of observing z when moving to $(s', G', \theta'_{G'})$ by doing action a. Here we simply observe the state of the MDP, so $O(s', G', \theta'_{G'}, a, z) = I_{s'}(z)$.
- $R' : S' \times A' \to \mathbb{R}$, the reward function, which corresponds directly to the rewards obtained in the MDP, i.e. $R'(s, G, \theta_G, a) = R(s, a)$.

Given that the state is not directly observable (i.e. we do not know the correct structure and parameters), we maintain a probability distribution over states, called a *belief*. The initial belief state in this POMDP is the initial state of the environment, along with priors $P(G_a, \theta_{G_a}), \forall a \in A$. At time t, the belief state corresponds to the current state of the MDP, s_t, along with posteriors $P(G_a, \theta_{G_a} | h_t), \forall a \in A$, where h_t is the history of actions and observations up to time t.

To represent this belief compactly, we assume that the joint priors $P(G_a, \theta_{G_a})$ satisfy the assumptions stated in section 2.2, namely they factorize into a product $P(G_a, \theta_{G_a}) = P(G_a)P(\theta_{G_a} | G_a)$ and the $P(\theta_{G_a} | G_a)$ are defined by a product of independent Dirichlet distributions. For each graph G_a, starting from prior counts $\phi_{G_a}^{i,v|E}$ for all state variables i, values $v \in S_i$, and parent values $E \in S_{\mathrm{Par}_i(G_a)}$, the posterior counts are maintained by simply incrementing by 1 the counts $\phi_{G_a}^{i, s'_i | \mathrm{ParVal}_i(s, G_a)}$ for all state variables i, each time a transition (s, a, s') occurs. As mentioned in section 2.2, the main difficulty is in maintaining the posterior $P(G_a | h)$, which is infeasible when the space of graphs is large. We approximate this using a particle filter, and for each particle (i.e. a sampled graph G_a), the posterior $P(\theta_{G_a} | G_a)$ is maintained exactly with counts ϕ_{G_a}. This particle filter is explained in more detail in the next section.

Finding the optimal policy for this POMDP yields an action selection strategy that optimally trades-off between exploration and exploitation such as to maximize long term expected return given the current model posterior and state of the agent. Our Bayesian RL approach therefore requires solving this POMDP. While many algorithms exist to solve POMDPs, few of them can handle high-dimensional infinite state spaces, as is required here. Hence, we propose to use online Monte Carlo methods to solve this challenging optimization problem (McAllester & Singh, 1999).

3.2 Online Monte Carlo Planning Algorithm

To solve the planning problem outlined above, we need efficient approximation methods, and in particular we turn to online sampling techniques to overcome the curse of dimensionality.

First, as mentioned above, we maintain the posterior $\Pr(G_a | h)$ using a particle filter algorithm. This is done by first sampling a set of K graphs from the prior $P(G_a)$ for each action a. We assign each graph a probability, $p_a^j = \frac{1}{K}$, for $j = 1 : K$. For each sampled graph, we also have a product of Dirichlet priors on the parameters θ_{G_a}. Whenever a transition (s, a, s') occurs, the probability p_a^j of graph G_a^j is updated:

$$\begin{aligned} p'^j_a &= \frac{1}{\eta} p_a^j \int P(s' | s, a, G_a^j, \theta_{G_a^j}) P(\theta_{G_a^j} | G_a^j, h) d\theta_{G_a^j} \qquad (8) \\ &= \frac{1}{\eta} p_a^j \prod_{i=1}^{n} \left[\phi_{G_a^j}^{i, s'_i | \mathrm{ParVal}_i(s, G_a^j)} / \sum_{v \in S_i} \phi_{G_a^j}^{i, v | \mathrm{ParVal}_i(s, G_a^j)} \right] \end{aligned}$$

where the integral term is just the expected probability of $P(s' | s, a)$ under the current posterior for $\theta_{G_a^j}$, and η is a normalization constant such that $\sum_{j=1}^{K} p'^j_a = 1$. For the Dirichlet posterior $P(\theta_{G_a^j} | G_a^j, h)$ associated with G_a^j, the appropriate counts are updated each time a corresponding state transition occurs.

Turning our attention to the planning problem, we now search for the best action to execute, given the current state, the current distribution on graphs (defined by p_a^j), and the current posterior over parameters for each graph. Define $Q^*(s, b, a)$ to be the maximum expected sum of rewards (i.e. the value) of applying action a when the agent is in MDP state s and has posterior b over DBNs. Then the optimal value is defined by

$V^*(s,b) = \max_{a \in A} Q^*(s,b,a)$ and the best action to apply is simply $\arg\max_{a \in A} Q^*(s,b,a)$.

Algorithm 1 $\mathrm{V}(s,b,d,N)$

```
if d = 0 then
    return V̂(s,b)
end if
maxQ ← −∞
for a ∈ A do
    q ← R(s,a)
    for j = 1 to N do
        Sample s' from P(s'|s,b,a)
        b' ← UpdateGraphPosterior(b,s,a,s')
        q ← q + (γ/N) V(s',b',d−1,N)
    end for
    if q > maxQ then
        maxQ ← q
        maxA ← a
    end if
end for
if d = D then
    bestA ← maxA
end if
return maxQ
```

A recursive approach for tractably estimating $V^*(s,b)$ using a depth-limited online Monte Carlo search is provided in Algorithm 1. Every time the agent needs to execute an action, the function $\mathrm{V}(s,b,D,N)$ is called for the current state s and posterior b. D corresponds to the depth of the search tree (i.e. planning horizon) and N to the branching factor (i.e. number of successor states to sample at each level, for each action). To sample a successor state s' from $P(s'|s,b,a)$, we can simply sample a graph G_a for action a according to the probabilities p_a^j and then sample s' from this DBN, given that the parents take values s. At the fringe, an estimate $\hat{V}(s,b)$ of the return obtained from this posterior is used. Several techniques can be used to estimate $\hat{V}(s,b)$. For instance one could maintain an approximate value function $\hat{V}_j(s)$ for each sampled factored MDP defined by the DBNs $\{(G_a^j, \phi_{G_a^j}) | a \in A\}$ and then compute $\hat{V}(s,b) = \sum_{j=1}^{K} \hat{V}_j(s) \prod_{a \in A} p_a^j$. The approximate value functions $\hat{V}_j(s)$ can be updated efficiently via prioritized sweeping every time the counts ϕ are updated. For the experiments presented below, we simply use $\hat{V}(s,b) = \max_{a \in A} R(s,a)$. The UpdateGraphPosterior updates the Dirichlet posteriors and probabilities p_a^j presuming a transition (s,a,s') was observed. The best action to execute for the current time-step can be retrieved through the $bestA$ variable for the top node of the tree. The computation time allowed to estimate $V^*(s,b)$ can be limited by controlling the branching factor (N) and search depth (D), albeit at the expense of lesser accuracy.

3.3 Resampling DBNs

The current approach is not particularly effective when the initial set of sampled DBN structures is poor, since we are simply updating weights and therefore not changing the structure. This can be addressed by resampling new DBNs from the current posterior $P(G|h)$ to obtain more likely structures after observation of the history h. We implement this using an MCMC algorithm, as described in section 2.2. In general, it may not be appropriate to re-sample graphs too frequently. One useful criteria to decide when to resample new graphs is to look at the overall likelihood L_a of our current set of DBNs for a particular action a. This can be computed directly from the normalization constant η (Equation 8). Presuming that at time $t = 0$, $L_a = 1$, we can simply update $L'_a = \eta L_a$ at every step. Then whenever L_a falls below some threshold, we resample a new set of K graph structures G'^j_a from posterior $P(G_a|h)$ and update the Dirichlet posterior $P(\theta_{G_a^j}|G_a^j, h)$ for each graph according to the whole history h (starting from the Dirichlet prior $P(\theta_{G_a}|G_a)$). The probabilities p_a^j for these new graphs are then reinitialized to $\frac{1}{K}$ and the likelihood L_a to 1.

4 Experiments

To validate our approach, we experiment with instances of the network administration domain (Guestrin et al., 2003). A network is composed of n computers linked together by some topology. Each computer is either in *running* or *failure* mode. A running computer has some probability of transitioning to failure, independent of its neighbors in the network; that probability is increased for every neighbor in failure mode. A computer in failure mode remains so until rebooted by the operator. A reward of +1 is obtained for every running computer in the network at every step, no reward is given for failed computers, and a -1 reward is received for each rebooting action. The goal of the operator is to maximize the number of running computers while minimizing reboots actions. The starting state assumes all computers are running.

In our experiments, we assume a probability $\frac{1}{30}$ that a running computer goes into failed mode and a probability $\frac{1}{10}$ that a failed computer induces failure in any of its neighbors. So at any step, the probability that a running computer remains in a running state is $\frac{29}{30}(0.9)^{N_F}$ where N_F is the number of neighbors in failure state. We assume a discount factor $\gamma = 0.95$.

This problem can be modeled by a factored MDP with n binary state variables, each representing the running state of a computer in the network. There are $n+1$ actions: a reboot action for each computer and

a *DoNothing* action. The DBN structure representing the dynamics when no reboot is performed is a bipartite graph where the state variable S'_i (the next state of computer i) depend on S_i (the previous state of computer i) and S_j for all computers j connected to i in the network. Note that if S'_i depends on S_j, then this implies j is connected to i and thus S'_j depends on S_i. Hence the adjacency matrix $\mathcal{A}$ encoding the dependence relations in this bipartite graph, where entry $\mathcal{A}_{ij} = 1$ if S'_j depend on S_i, 0 otherwise, is always symmetric and has a main diagonal full of ones.

In terms of prior knowledge, we assume the agent knows that rebooting a computer always puts it back into running mode and doesn't affect any other computer. The goal of the agent is to learn the behavior of each computer in the network when no reboot is performed on them. Therefore, a single DBN is learned for the behavior of the system when no reboot is performed. We also assume the agent knows that the adjacency matrix is symmetric and has a main diagonal of ones. However we do not assume that the agent knows the topology of the network. We choose a prior over structures that is a uniform distribution over bipartite graphs with symmetric adjacency matrix (and main diagonal equal to 1). Given a prior of this form, the set of moves we consider to sample graphs in the Metropolis-Hasting algorithm consist of inverting any of the binary variables in the upper-right half of the adjacency matrix $\mathcal{A}$ (excluding the main diagonal) as well as the corresponding entry in the bottom-left half. Moves of this type preserve the symmetry in the adjacency matrix, and correspond to adding or removing a connection between any pair of computers in the network. We assume no prior knowledge regarding the probabilities of failure, so a uniform Dirichlet prior was used. Under the likelihood equivalence assumption, the prior counts ϕ_G are defined such that $\phi_G^{i,v|E} = \frac{1}{|S_{\mathrm{Par}_i(G)}||S_i|} = 2^{-|\mathrm{Par}_i(G)|-1}$.

We consider three different network architectures: a simple linear network of 10 computers (1024 states), a ternary tree network composed of 13 computers (8192 states) and a dense network of 12 computers (4096 states) composed of 2 fully connected components of 6 computers, linked to each other. These networks are shown in Figure 1. To assess the performance of our structured Bayesian RL approach, we compare it to a similar model-based Bayesian RL that learns the full joint distribution table, i.e. the DBN where each next state variable S'_i depends on all previous state variables S_j. We also consider the case where the DBN structure is fully known in advance and only the probability parameters are learned. These three approaches are compared in terms of three different metrics:

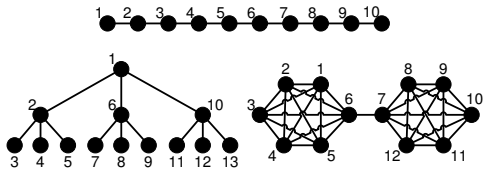


Figure 1: Linear network (top), ternary tree network (left) dense network (right).

empirical return, distribution error and structure error, as a function of the number of learning steps. The distribution error corresponds to a weighted sum of L1-distance between the distributions of the next state variables as defined by the Dirichlet posterior counts and the exact distributions in the system: $\sum_{j=1}^{K} p_a^j \sum_{s\in S} \sum_{i=1}^{n} \left|\left| \frac{\phi^{i,*|\mathrm{ParVal}_i(s,G_a^j)}_{G_a^j}}{||\phi^{i,*|\mathrm{ParVal}_i(s,G_a^j)}_{G_a^j}||_1} - P(S'_i|s,a) \right|\right|_1$. The structure error is computed as a weighted sum of the errors in the adjacency matrix of the sampled graphs compared to the correct adjacency matrix: $\sum_{j=1}^{K} p_a^j \sum_{i=1}^{n} \sum_{k=1}^{n} |\mathcal{A}_{ik}^{G_a^j} - \mathcal{A}_{ik}^{G^*}|$, where $\mathcal{A}^{G_a^j}$ is the adjacency matrix for sampled graph G_a^j and $\mathcal{A}^{G^*}$ the exact adjacency matrix. All reported results are averaged over 50 simulations of 1500 steps each. Error bars were small, so were removed for clarity.

4.1 Linear Network

In the linear network experiment, we sample $K = 10$ graphs, and resampling is performed whenever $\ln L_a < -100$. Online planning is done with depth $D = 2$ and branching factor $N = 5$ for each action. Since we use the immediate reward at the fringe of the search tree, this corresponds to approximate planning over a 3-step horizon. These same parameters are also used for planning with the known structure, and over the full joint probability table. Results are presented in Figures 3-5.

These figures show that our approach (denoted *Structure Learning*) obtains similar returns as when the structure is known in advance (denoted *Known Structure*). Both of these cases reach optimal return (denoted *Known MDP*[1]) very quickly, within 200 steps. Our approach is also able to learn the transition dynamics as fast as when the structure is known a priori. On the other hand, the unstructured approach (denoted *Full Joint*) takes much more time to achieve a good return and learn the dynamics of the system. This confirms that assuming a structured representation of the system can significantly speed up learning. Finally, we also observe that the structure learning al-

[1]This is the value iteration solution, assuming the structure and parameters are fully known in advance.

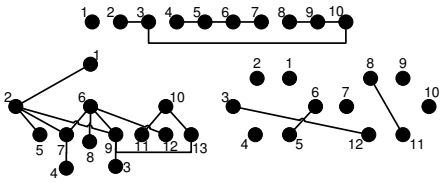


Figure 2: Most likely networks among samples after 1500 steps: Linear network (top), ternary tree network (left) dense network (right).

gorithm is able to learn a good structure of the domain over time (see Figure 4.1). Even though the sampled structures are not perfect, our approach is still able to predict future states of the system with similar accuracy as when the structure is known in advance. The average planning times per action are 100ms for structure learning, and 19ms for the other two approaches with fixed structure.

4.2 Ternary Tree Network

In the ternary tree network experiment, we sample $K = 8$ graphs, and resample them whenever $\ln L_a < -150$. For the planning, we use a depth $D = 2$ and sample $N = 4$ next states for each action. Results are presented in Figures 6-8. The results are similar to the Linear Network experiment. The main point to note is that this is a significantly harder problem for the unstructured approach, which even after 1500 steps of learning has not yet improved. This is in contrast to our approach which obtains similar performance as when the structure is known a priori, and reaches optimal performance after just a few hundred steps of learning. These results are obtained even though the priors we provide are very weak. The average planning times per action are 153ms for structure learning, and 29ms for the two approaches with fixed structure.

4.3 Dense Network

In the dense network experiment, we sample $K = 8$ graphs, and resample them whenever $\ln L_a < -120$. For the planning, we assume $D = 2$ and $N = 4$. Results are presented in Figures 9-11. In this domain, we observe a surprising result: our approach using structure learning is able to learn the dynamics of the system much faster than when the structure is known in advance (see Figure 10), even though the learned structures are still far from correct (see Figures 11 and 4.1). This is a domain where there are many dependencies between state variables, so there are many parameters to learn (whether or not the structure is known). In such a case, our structure learning approach is at an advantage, because early on in the learning, it can favor simpler structures which approximate the dynamics reasonably well from very few learning samples (e.g. < 250). As further data is acquired, more complex structures can be inferred (and more parameters estimated), in which case our approach achieves similar return as when the structure is known, while it continues to estimate the true parameters more accurately.

This result has important implications for RL in large domains. Namely, it suggests that even in domains where significant dependencies exist between state variables, or where there is no apparent structure, a structure learning approach can be better than assuming a known (correct) structure, as it will find simple models that allow powerful generalization across similar parameters, thus allowing for better planning with only a small amount of data.

The average planning times per action are 120ms for structure learning, and 22ms for the other two approaches with fixed structure.

5 Conclusion

This paper presents a novel Bayesian framework for learning both the structure and parameters of a factored MDP, while also simultaneously optimizing the choice of actions to trade-off between model exploration and exploitation. It is important to note that both the use of a factored representation, and the use of online planning, are key to allowing our approach to scale to large domains. By learning a factored representation, we allow powerful generalization between states sharing similar features, hence learning of the model makes more efficient use of data. It is especially interesting to notice that our structure learning approach is a useful way to accelerate RL even in domains with very weak structure.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT).

References

Bellman, R. (1957). A markovian decision process. *Journal of Mathematics and Mechanics*, *6*.

Boutilier, C., Dearden, R., & Goldszmidt, M. (2000). Stochastic dynamic programming with factored representations. *Artif. Intel.*, *121*(1-2), 49–107.

Dearden, R., Friedman, N., & Andre, D. (1999). Model based bayesian exploration. In *UAI*, pp. 150–159.

Duff, M. (2002). *Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes*. Ph.D. thesis, University of Massassachusetts Amherst.

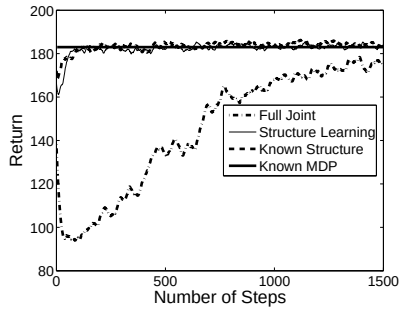


Figure 3: Empirical return in the linear network.

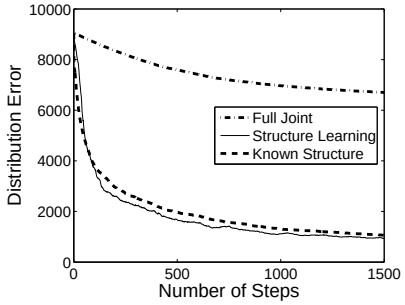


Figure 4: Distribution error in the linear network.

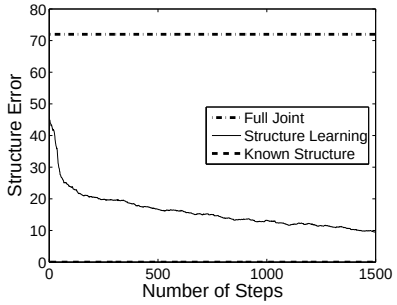


Figure 5: Structure error in the linear network.

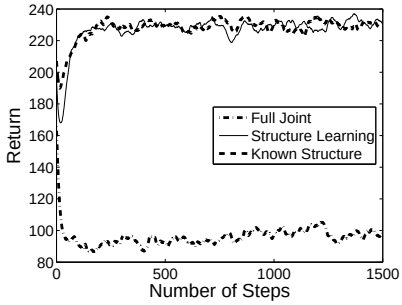


Figure 6: Empirical return in the ternary tree network.

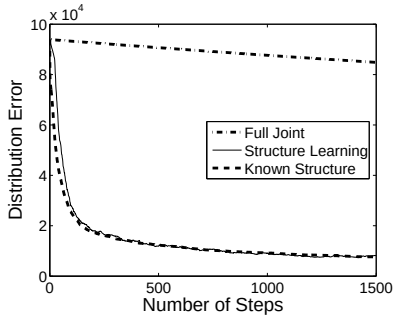


Figure 7: Distribution error in the ternary tree network.

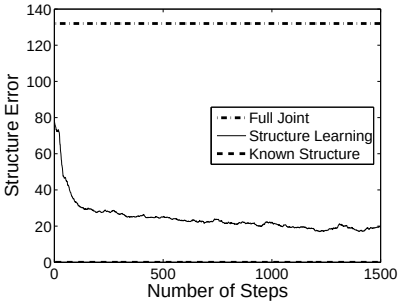


Figure 8: Structure error in the ternary tree network.

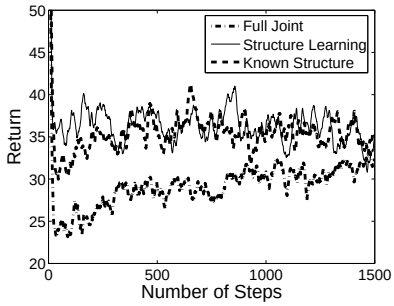


Figure 9: Empirical return in the dense network.

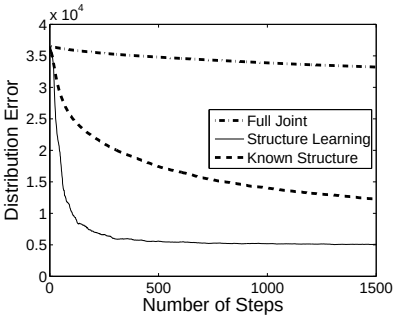


Figure 10: Distribution error in the dense network.

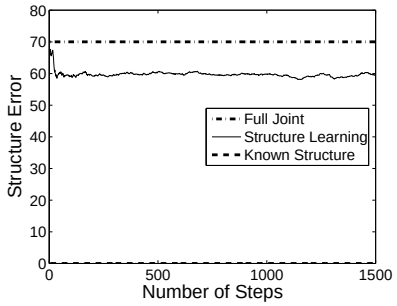


Figure 11: Structure error in the dense network.

Eaton, D., & Murphy, K. (2007). Bayesian structure learning using dynamic programming and MCMC. In *UAI*.

Friedman, N., & Koller, D. (2003). Being Bayesian about Bayesian network structure: A Bayesian approach to structure discovery in Bayesian networks.. *Machine Learning*, *50*(1–2), 95–125.

Guestrin, C., Koller, D., Parr, R., & Venkataraman, S. (2003). Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, *19*, 399–468.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, *20*(3), 197–243.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*(1-2), 99–134.

McAllester, D., & Singh, S. (1999). Approximate Planning for Factored POMDPs using Belief State Simplification. In *UAI*, pp. 409–416.

Poupart, P., Vlassis, N., Hoey, J., & Regan, K. (2006). An analytic solution to discrete Bayesian reinforcement learning. In *ICML*, pp. 697–704.

Sutton, R., & Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

CT-NOR: Representing and Reasoning About Events in Continuous Time

Aleksandr Simma, Moises Goldszmidt, John MacCormick, Paul Barham, Richard Black, Rebecca Isaacs, Richard Mortier*
Microsoft Research

Abstract

We present a generative model for representing and reasoning about the relationships among events in continuous time. We apply the model to the domain of networked and distributed computing environments where we fit the parameters of the model from timestamp observations, and then use hypothesis testing to discover dependencies between the events and changes in behavior for monitoring and diagnosis. After introducing the model, we present an EM algorithm for fitting the parameters and then present the hypothesis testing approach for both dependence discovery and change-point detection. We validate the approach for both tasks using real data from a trace of network events at Microsoft Research Cambridge. Finally, we formalize the relationship between the proposed model and the noisy-or gate for cases when time can be discretized.

1 Introduction

The research described in this paper was motivated by the following real life application in the domain of networked distributed systems: In a modern enterprise network of scale, dependencies between hosts and network services are surprisingly complex, typically undocumented, and rarely static. Even though network management and troubleshooting rely on this information, automated discovery and monitoring of these dependencies remains an unsolved problem. In [2] we described a system called *Constellation* in which computers on the network cooperate to make this information available to all users of the network. Constellation takes a black-box approach to locally (at each computer/server in the network) learn explicit dependencies between its services using little more than the timings of packet transmission and reception. The black-box approach is necessary since any more processing of the incoming and outgoing communication packages would imply prohibitive amounts of overhead on the computer/server. The local models of dependency can then be recursively and distributively composed to provide a view of the global dependencies. In Constellation, computers on the network cooperate to make this information available to all users in the network.

Constellation and its application to system wide tasks such as characterizing a networking site service and hosts dependencies for name resolution, web browsing, email, printing, reconfiguration planning and end-user diagnosis are described in [2]. This paper focuses on the probabilistic and statistical building blocks of that system: the probabilistic model used in the local learning, the EM algorithm used to fit the parameters of the model, and the statistics of the hypothesis testing used to determine the local dependencies. The model, which we call *Continuous Time Noisy Or* (CT-NOR), takes as input sequences of input events and output events and their time stamps. It then models the interactions between the input events and output events as Poisson processes whose intensities are modulated by a (parameterized) function taking into account the distance in time between the input and output events. Through this function the domain expert is able to explicitly encode knowledge about the domain. The paper makes the following contributions:

* John is now with Dickinson College, PA. Work done while a Researcher with Microsoft Research. Alex is with the University of California, Berkeley, CA. Work done while an intern with Microsoft Research.

1. Develops an EM algorithm for fitting all the parameters of this model and an algorithm for dependence discovery and change point detection based on statistical hypothesis testing.

2. Evaluates the performance of the model and the inference procedures both on synthetic data and on real life data taken from a substantial trace of a large computer network.

3. Formalizes the relationship between CT-NOR and the noisy-or (NOR) gate [11] when the time between the events can be discretized.

This paper is organized as follows: Section 2 describes the model and Section 3 describes the EM algorithm for fitting the parameters. Section 4 is concerned with the relation to the NOR gate. The algorithms and framework for applying the model to dependency discovery and change point detection is described in Section 5. That section also contains validation experiments with synthetic data. Section 6 contains experiments on real data and results. Finally, Section 7 has some conclusions and future work.

2 The CT-NOR model

In this section we formally describe the CT-NOR model with the objective of building the likelihood equation. First, we provide some background on Poisson Processes, and then we use them to construct the model (Eq. 4).

A Poisson Process[1] can be thought of as random process, samples from which take the form of a set of times at which events occurred. A Poisson Process is defined over a mean (base) measure $f(t)$ and is characterized the property that for any interval (t_1, t_2), the number of events that occur in that interval follows the Poisson distribution with the parameter $\int_{t_1}^{t_2} f(t)dt$. Furthermore, the number of events that occur on two disjoint intervals are independent.

Let us use "channel" to denote a sequence of events.[2] The CT-NOR model considers a single output channel and a set of input channels. Let o_l denote the time of the lth output event and $i_k^{(j)}$ the time of the kth input on channel j. Furthermore, let n denote the number of output events and $n^{(j)}$ the number of input events on channel j. Then event k in input channel j *generates* a Poisson process of output events with the base measure $p_k^{(j)}(t) = w^{(j)} f_\theta(t - i_k^{(j)})$.

The term $w^{(j)}$ represents the average number of output events that we expect each input event on channel j to be responsible for, and $f_\theta(t)$ is the distribution of the delay between an input and the output events caused by it, taking as its argument the delay between the time of the output o_l and the time of the input $i_k^{(j)}$. The mathematical structure of the intensity makes intuitive sense: the probability that a given input event caused a given output event depends on both the expected number of events it generates and the "distance" in time between them.

We recall that given multiple independent Poisson processes (denoted as PP) we can use the sum of their intensities to construct a "global" Poisson process and write $\{o_l\} \sim PP(\sum_j \sum_{k=1}^{n^{(j)}} p_k^{(j)}(t))$ as the probability of the set of n outputs $\{o_l\}$, $1 \leq l \leq n$. The double sum runs over all the channels and over all input events in the channels. Intuitively, and similar to the NOR gate in graphical models [11], the independence between the between input channels translates into a model where the events in the output channel are "caused" by the presence of any (a disjunction) of input events in the input channels (with some uncertainty). The formal relation with NOR is presented in Section 4.

We now proceed to write the likelihood of the data given the model and the input events. Let $\lambda = \sum_j n^{(j)} w^{(j)}$, the total mass of the Poisson base measure. The number n of outputs is distributed as a Poisson distribution

$$n \sim Poisson(\lambda), \tag{1}$$

and the location of a specific output event o_l is distributed with the probability density

$$o_l \sim \frac{\sum_j \sum_{k=1}^{n^{(j)}} p_k^{(j)}(o_l)}{\lambda} \text{ for } l = 1 \ldots n \tag{2}$$

$$= \frac{\sum_j \sum_{k=1}^{n^{(j)}} w^{(j)} f_\theta(o_l - i_k^{(j)})}{\lambda} \tag{3}$$

[1] This overview is very informal. The more general and formal measure-theoretic definition can be found in [5].

[2] In the domain of computer networks, a channel refers to a unidirectional flow of networked packets. Thus a channel will be identified by the service (e.g., HTTP, LDAP, etc) and the *IP* address of the source or destination. In this paper we identified the packets with events as it is only their time stamp that matters.

The likelihood of observing a set $\{o_l\}$ of outputs is[3]:

$$L(o|i) = \lambda^n \cdot e^{-\lambda} \prod_{l=1}^{n} \sum_{jk} \frac{w^{(j)} f_\theta(o_l - i_k^{(j)})}{\lambda} \qquad (4)$$

Before concluding this section, we expand a bit on the function f_θ as it is an important part of the model. This function provides us with the opportunity of encoding domain knowledge regarding the expected shape of the delay between input and output events. In our experience using CT-NOR to model an enterprise network we used two specific instantiations: a mixture of a narrow uniform and a decaying exponential and a mixture of a uniform and Gaussian. The uniform distribution captures the expert knowledge that a lot of the protocols involve a response within a window of time (we call this co-occurrence). The Gaussian delay distribution extends the intuitions of co-occurrence within a window to also capture dependencies that can be relatively far away in time (such as with the printer). The left tail of the Gaussian corresponding to negative delays is truncated. The exponential distribution captures the intuition that the possibility of dependency decays as the events are further away in time (this is true for the HTTP protocol). We will not explicitly expand these functions in the derivations as they tend to obscure the exposition. Needless to say that the parameters of these functions are all fitted automatically using EM as described in the next section.

Groups of channels may have different delay distributions, in which case the delay distribution can be indexed by the channel group and all the derivations in this paper remain the same. For example, channels can be grouped by network service, where all HTTP channels have the same delay distribution (thus allowing data from multiple channels to assist in parameter fitting), but the DNS channels are allowed a different delay distribution. All the experiments in the paper use a *leak* — a pseudo-channel with a single event at the start of the observation period and a delay distribution that is uniform over the length of the observations. This leak captures events which are not explained by the remaining channels.

[3]Since the Poisson Process produces unordered outputs but the events are considered to be sorted, a permutation factor of $n!$ is required. It cancels out the $n!$ in the Poisson density.

3 Fitting a CT-NOR model

We perform inference and estimation on the model through the EM algorithm. We first set the stage by finding a suitable bounding function $B(z)$ for the likelihood. The EM algorithm iteratively chooses a tight bound in the E step and then maximizes the bound in the M step. Let $z_{kl}^{(j)}$ be some positive vector such that $\sum_{jk} z_{kl}^{(j)} = 1$ for each l. For a fixed l, $z_{kl}^{(j)}$ is the probability of the latent state indicating that packet k on channel j caused output l. Then from Eq. 4:

$$\begin{aligned}
\log L(o|i) &= -\lambda + \sum_{l=1}^{n} \log \sum_{jk} w^{(j)} f_\theta(o_l - i_k^{(j)}) \\
&= -\lambda + \sum_{l=1}^{n} \log \sum_{jk} z_{kl}^{(j)} \frac{w^{(j)} f_\theta(o_l - i_k^{(j)})}{z_{kl}^{(j)}} \\
&= -\lambda + \sum_{l=1}^{n} \log \mathbb{E}_z \frac{w^{(j)} f_\theta(o_l - i_k^{(j)})}{z_{kl}^{(j)}}
\end{aligned}$$

Now, by Jensen's inequality, $\log L(o|i) \geq B(z)$ where:

$$B(z) = -\lambda + \sum_{l} \mathbb{E}_z \log \frac{w^{(j)} f_\theta(o_l - i_k^{(j)})}{z_{kl}^{(j)}}$$

3.1 E-Step

For a particular choice of θ (the parameters of the f_θ function) and $w^{(j)}$, the bound above is tight when

$$z_{kl}^{(j)} = \frac{w^{(j)} f_\theta(o_l - i_k^{(j)})}{\sum_{j'k'} w^{(j')} f_\theta(o_l - i_{k'}^{(j')})}$$

because in that case, $\frac{w^{(j)} f_\theta(o_l - i_k^{(j)})}{z_{kl}^{(j)}}$ is a constant for a fixed l and $\mathbb{E} \log C = \log \mathbb{E} C = \log C$. Therefore, we use these choice of $z_{kl}^{(j)}$.

3.2 M-step

For a fixed choice of $z_{kl}^{(j)}$, we need to maximize the bound with respect to $w^{(j)}$ and θ.

Optimizing with respect to $w^{(j)}$, we notice that the derivative is

$$\frac{\partial B}{\partial w^{(j)}} = -n^{(j)} + \sum_{l} \sum_{k} z_{kl}^{(j)} \frac{1}{w^{(j)}}$$

yielding

$$\hat{w}^{(j)} = \frac{\sum_{kl} z_{kl}^{(j)}}{n^{(j)}}$$

With respect to θ, we can say that

$$\hat{\theta} \quad = \quad \arg\max_{\theta} \sum_{jkl} z_{kl}^{(j)} \log f_\theta(o_l - i_k^{(j)})$$

which is simply the parts of the objective function that depend on θ. This can be a very easy optimization problem for a large class of distributions, as it is of the same form as maximum likelihood parameter observation given observed data points and corresponding counts. For example, for the exponential family, this simply requires moment matching: $\mu(\hat{\theta}) = \frac{\sum_{jkl} z_{kl}^{(j)} T(o_l - i_k^{(j)})}{\sum_{jkl} z_{kl}^{(j)}}$ where $\mu(\hat{\theta})$ is the mean parameterization of the estimated parameter $\hat{\theta}$ and $T(\cdot)$ are the sufficient statistics for the family.

4 Relation to Noisy Or

As an alternative model, consider binning the observed data into windows of width δ and modeling the presence or absence of output events in a particular bin as a NOR [11]. The possible explanations (parents) are the presence of input events in preceding windows. We will show that a particular, natural parameterization of the NOR model is equivalent to CT-NOR in the limit, as the bin width approaches zero. This relationship is important because it provides a nontrivial extension of NOR to domains with continuous time and provides insight into the independence structure of the two models.

Let $\mathbb{O}_t^\delta$ be an indicator of presence of output events between the times $t\delta$ and $t\delta + \delta$ and $\mathbb{I}_t^{(j)\delta}$ be the indicator for input events from channel j in that same time period. We will use P_{NOR} to denote the probability under the NOR model and $P_{\text{CT-NOR}}$ for probability under CT-NOR.

$$P_{\text{NOR}}(\mathbb{O}_t^\delta = 0|\text{Input}) \quad = \quad \prod_j \prod_{s<t} (1 - p_{(t-s)}^{(j)} \mathbb{I}_s^{(j)\delta})$$

The $p_{(t-s)}^{(j)}$ is the weight associated with the possible explanation $\mathbb{I}_s^{(j)\delta}$. To prevent the number of parameters from increasing as the bin size becomes small, reparameterize with

$$p_{(t-s)}^{(j)} = w^{(j)} f_\theta(\delta(t-s))\delta$$

for any distribution f_θ that satisfies some technical conditions.[4] Since f_θ may be a very flexible family of distributions, this parameterization imposes only minor constraints on the weights, but will be useful for reasoning about NOR models which model the same data but with differing bin widths. When the bin width is halved, the probability that one of the sub-bins has an output event must be equal to the probability that the large bin has an output event plus a second-order term. This condition is required for a coherent parameterization of a family of NOR distributions and follows from the technical conditions placed on f_θ.

We argue that as the bin width δ decreases, this model becomes equivalent to a CT-NOR with a suitable choice of parameters. Choose a δ sufficiently small that each bin contains at most one input event per channel, and at most one output event. We will use P_{NOR}^t to denote $P_{\text{NOR}}(\mathbb{O}_t^\delta = 0|\text{Input})$, the probability that the tth bin has no output events falling into it.

$$\begin{aligned} P_{\text{NOR}}^t &= \prod_j \prod_{s<t} \left(1 - w^{(j)} f_\theta((t-s)\delta)\mathbb{I}_s^{(j)\delta}\right) \\ &= \prod_j \prod_k \left(1 - w^{(j)} f_\theta(t\delta - i_k^{(j)})\delta + o(\delta^2)\right) \\ &= 1 - \delta \sum_j \sum_k \left(w^{(j)} f_\theta(t\delta - i_k^{(j)})\right) + o(\delta^2) \end{aligned}$$

Under a CT-NOR model which uses the same $w^{(j)}$ and the same f_θ, the probability of not observing any outputs is very similar. We use π to denote the parameter of the Poisson random variable governing the number of outputs in the interval.

$$\begin{aligned} \pi &= \sum_j \sum_k w^{(j)} \int_{t\delta}^{t\delta+\delta} f_\theta(x - i_k^{(j)})dx \\ &= \delta \sum_j \sum_k w^{(j)} f_\theta(t\delta - i_k^{(j)}) + o(\delta^2) \\ P_{\text{CT-NOR}}^t &= P\left[\text{Poisson}(\pi) = 0\right] \\ &= \exp(-\pi) \\ &= 1 - \pi + o(\delta^2) \\ &= P_{\text{NOR}}^t + o(\delta^2) \end{aligned}$$

[4] It is sufficient for the density to exist and be Lipschitz, which means that there exists a constant C such that $|f_\theta(a) - f_\theta(b)| \leq C|a - b|$ for any a, b. Any continuously differentiable function with a bounded derivative satisfies this condition. It is easy to extend this proof to any bounded density with a finite number of discontinuities which has a bounded derivative everywhere except for the discontinuities.

These results can be combined to demonstrate that the probability assigned to any set of output events by the two models is equal up a factor of $(1+o(n\delta))$ which converges to 1 as δ decreases to zero. The asymptotics are in terms of bin width δ decreasing to zero for a fixed set of observations, so n and T are constant.

$$\begin{aligned}
&\frac{P_{\text{NOR}}(\text{Out}|\text{In})}{P_{\text{CT-NOR}}(\text{Out}|\text{In})} \\
&\quad = \prod_{t=0}^{T/\delta} \left(\frac{P^t_{\text{NOR}}}{P^t_{\text{CT-NOR}}}\right)^{1-\mathbb{O}^\delta_t} \left(\frac{1-P^t_{\text{NOR}}}{1-P^t_{\text{CT-NOR}}}\right)^{\mathbb{O}^\delta_t} \\
&\quad = (1+o(\delta^2))^{T/\delta-n} \cdot (1+o(\delta))^n \\
&\quad = (1+(T/\delta-n)o(\delta^2)) \cdot (1+no(\delta)) \\
&\quad = (1+(T+n)o(\delta)) \\
&\quad = (1+o(\delta))
\end{aligned}$$

CT-NOR and NOR with an increasingly small bin size assign equivalent probability to any sequence of output events, indicating that the two classes of models are closely related, and that CT-NOR is the model that emerges as the limit when the NOR's bin size is decreased toward zero.

5 Dependence discovery and change point detection

With the probabilistic framework described in the previous section, we can use statistical machinery to perform inference for two applications: a) input-output relation discovery and b) change-point detection. The next two subsections describe the algorithms in detail and also validate the main assumptions using synthetically generated data. The final subsection (5.3) describes a computationally efficient approximation to the hypothesis test procedures.

5.1 Dependence discovery

For the purposes of network management, a crucial problem is dependence discovery. For each computer in the network, we are interested in automatically finding out from observations which input channels have a causal effect on an output channel.

We can frame the dependency discovery task as hypothesis testing. Specifically, testing whether an input channel j causes output events corresponds to testing the hypothesis that $w^{(j)} = 0$. One way of testing this hypothesis is through the likelihood ratio test [14]. We fit two models: M_{full}, under which, all the parameters are unrestricted, and M_{res}, under which $w^{(j)}$ is constrained to be zero. The test statistic in this case is

$$-2\log\Lambda = -2\log\frac{L_{M_{\text{res}}}(\text{Data})}{L_{M_{\text{full}}}(\text{Data})}$$

The asymptotic distribution of this test statistic is called a $\bar{\chi}^2$ and is a mixture of χ^2 with different degrees of freedom. The weights depend on the Fisher information matrix and are difficult to compute[7], but the significant terms in the mixture are χ^2_1 and χ^2_0 which is a delta function at zero. The $\bar{\chi}^2$ emerges as the null distribution instead of the more familiar χ^2 because the weight parameters $w^{(\cdot)}$ are constrained to be non-negative, and when an estimated $\hat{w}^{(j)}$ is zero in the unconstrained model, imposing the constraint does not change the likelihood. If a set of true null hypotheses is known, the mixture coefficients can be trivially estimated, with the weight of χ^2_0 being the proportion of test statistics that are 0. When no ground truth is available, the proportion of null hypotheses can be estimated using the method described in [13] and then used to estimate the mixture proportions.

To demonstrate that the model efficiently recovers the true causal channels and has the proper test-statistic distribution under the null hypothesis, we first test the model on synthetic data that is generated according to some instantiation of the model. 10 input channels are generated; half of them have no causal impact on output events and half produce a Poisson(0.01) number of output events with the delay distribution of Exponential(0.1). Note that the causality is weak – very few input events actively produce an output. For each hour, 500 input events per channel, the corresponding output events, and 100 uniformly random noise events (which are not caused by any input activity) are produced. The resulting p-values are plotted in Figure 1.

Observe that the null p-values (conditioned on the test statistic being non-zero) are distributed uniformly. This is evidenced by the p-values following the diagonal on the quantile-quantile plot. The alternative p-values (without any conditioning) for channels which exhibit causality are mostly very low, with 88% being below 0.1. Furthermore, the specific parameter estimates (the delay distribution parameter and $w^{(j)}$) are in line with their true values.

5.2 Changepoint Detection

When the relationship between events is altered, it can be an indication of a significant change in the

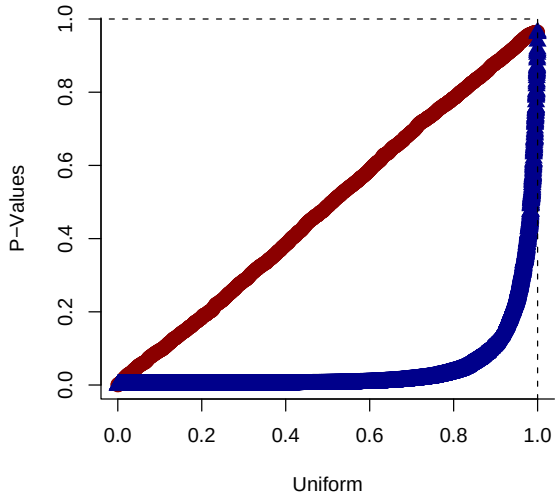


Figure 1: Quantile-quantile plot of dependency discovery p-values for 2 hours of synthetic data. The red circles are the distribution of p-values for the null hypotheses, and are uniform. The blue triangles show p-values of the alternative hypotheses and are small, indicating power.

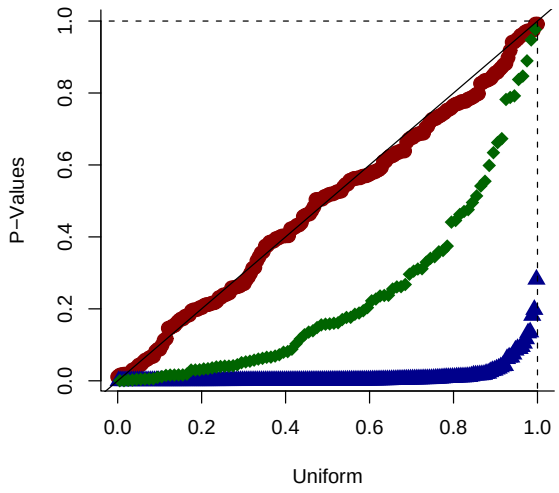


Figure 2: Quantile-quantile plot of the p-values for changepoint detection on synthetic data. The red circles are null hypotheses (no changepoint), the green diamonds are a weak alternative ($w^{(j)}$ increases from 0.01 to 0.02) and the blue triangles are a strong alternative ($w^{(j)}$ increases from 0.01 to 0.05).

system; in the case of Constellation, this is of interest to the system administrators. We describe a building block for identifying whether the parameters $w^{(j)}$ change between two time periods and demonstrate its correct functionality. Changepoint algorithms have long been studied in machine learning and statistics, and our test for whether the behavior of a parameter is altered between two time periods can be plugged into one of many existing algorithms. Furthermore, the simple two-period test described here is sufficient for many monitoring applications.

We again use the log-likelihood ratio test methodology. In order to do that, it is necessary to extend the model to allow the parameters to depend on time. The model can be written as

$$\{o\} \sim PP\left(\sum_j \sum_k w^{(j)}_{i^{(j)}_k} f_\theta(o_l - i^{(j)}_k)\right).$$

Detecting changepoints is accomplished by testing two hypotheses. The null is that the weights do not change between two time periods, and can be written as $w^{(j)}_t = w^{(j)}$. Under the alternative, for a particular channel of interest m and an interval of time S, the weight changes:

$$\begin{aligned} \forall j \neq m \quad w^{(j)}_t &= w^{(j)} \\ w^{(m)}_t &= w^{(m)} \text{ if } t \in S, w'^{(m)} \text{otherwise.} \end{aligned}$$

The existence of a changepoint is equivalent to rejecting the null hypothesis. Fitting the alternative model is a simple modification of the EM procedure described for the null model; for fast performance, it is possible to initialize at the null model's parameter values and take a single M step, reusing the latent variable distribution estimated in the E step. The test statistic in this case will again be $-2\log\Lambda$ and its null distribution will be χ^2 if the true $w^{(m)} > 0$ and $\bar{\chi}^2$ otherwise.

Figure 2 shows a quantile-quantile plot of the p-values (computed using the χ^2 distribution) under the null hypothesis, computed for causal channels of the same synthetic data as in section 5.1; there are two hours of data with 500 input events per channel per hour. As expected, the quantile-quantile plot forms a straight line, demonstrating that on the synthetic dataset, the null test statistic has a χ^2 distribution. When a strong changepoint is observed ($w^{(j)}$ changes from 0.01 to 0.05) , the p-values are very low. When a weak changepoint is observed ($w^{(j)}$ changes

from 0.01 to 0.02) the p-values are lower than under the null distribution but power is significantly lower than when detecting the major changepoint.

5.3 Bounding the log-likelihood ratio

Computing the log-likelihood ratio requires refitting a restricted model, though only a small number of EM steps is typically required. However, it is possible to bound the log likelihood ratio for dependency discovery very efficiently.

For the restricted model testing channel m's causality, we must compute the likelihood under the constraint that $w^{(m)} = 0$. Take the estimates of w of the unrestricted model and let $\alpha = \frac{\lambda}{\lambda - w^{(m)} n^{(m)}}$. Instead of computing the ratio with the true maximum likelihood parameters for the restricted model, we propose a set of restricted parameters, and compute the ratio using them. We produce a restricted version of parameters $w^{(\cdot)}$ by setting $w^{(m)}$ to zero and inflating the rest by a factor of α. That simply corresponds to imposing the restriction, and redistributing the weight among the rest of the parameters, so that the expected number of output packets remains the same. In that case,

$$
\begin{aligned}
-2\log\Lambda &= -2\log\frac{L_{M_{res}}(Data)}{L_{M_{full}}(Data)} \\
&\geq -2\log\prod_l \frac{\sum_{j\neq m,k} \alpha w^{(j)} f_\theta(o_l - i_k^{(j)})}{\sum_{jk} w^{(j)} f_\theta(o_l - i_k^{(j)})} \\
&= -2\log\prod_l \alpha\left(1 - \frac{\sum_k w^{(m)} f_\theta(o_l - i_k^{(m)})}{\sum_{jk} w^{(j)} f_\theta(o_l - i_k^{(j)})}\right) \\
&= -2\log\prod_l \alpha\left(1 - \sum_k z_{ml}^{(j)}\right)
\end{aligned}
$$

As a reminder, z_{ml}^{j} is the latent variable distribution estimated in the E-step of EM. Since the numerator of the log-likelihood ratio is a lower bound and the denominator exact, this expression is a lower bound on Λ. Intuitively, $\log\prod_l \left(1 - \sum_k z_{ml}^{(j)}\right)$ corresponds to the probability that channel m has exactly 0 output events assigned to it when causality is assigned according to the EM distribution on the latent variables . The $\log\alpha$ term corresponds to the increase in likelihood from redistributing channel m's weight among the other channels.

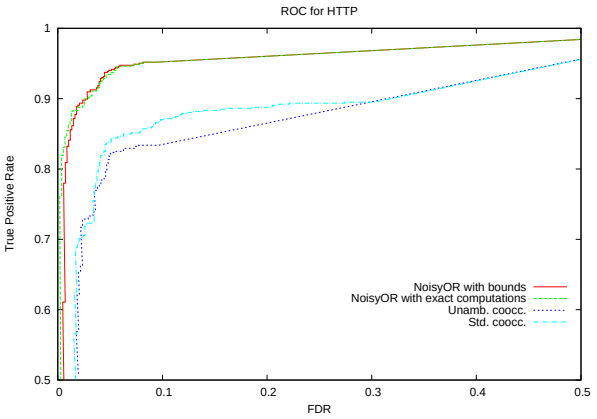


Figure 3: ROC for CT-NOR and competing algorithms on data from a real enterprise network. Both the exact an approximate CT-NOR tests produce detection results superior to the alternative methods.

6 Results

We describe the results of applying the algorithms of the previous section to a subset of a real dataset consisting of a trace comprising headers and partial payload of around 13 billions packets collected over a 3.5 week period in 2005 at Microsoft Research in Cambridge, England. This site contains about 500 networked machines and the trace captures conversations over 2800 off-site IP addresses. Ground-truth for dependence discovery and change point detection is not readily available and it has to be manually generated. We took 24 hours of data at the web proxy and manually extracted ground truth for the HTTP traffic at this server by deep inspection of HTTP packets. It is with this part of the data that we validate our algorithms, as it provides us with objective metrics, such as precision and recall, to assess the performance of our algorithms.

6.1 Dependency Discovery

First, we are interested in assessing the performance of the dependence discovery capabilities of our model and hypothesis testing algorithm. In the application of diagnosis and monitoring of networked systems it is crucial to maintain a consistent map of all the server and services inter-dependencies and their changes. Finding dependencies at the server level is the main building block used by Constellation [2] in building this global map. We compare our method to two other alternatives. One is a simple binomial test: for each input channel, we count the number of output packets falling within a W width window of

an input packet, and determine whether that number is significantly higher than if the output packets were uniformly distributed. We call this "standard co-occurrence." The second alternative considers an input and output channel to be dependent only if there is a unique input packet in the immediate vicinity of an output packet. The reason we select these two alternatives is that a) they reflect (by and large) current heuristics used in the systems community [1] and b) they will capture essentially the "easy" dependencies (as our results indicate).[5]

As can be seen on the ROC curve in Figure 3, CT-NOR successfully captures 85% of the true correlations with a 1% false positive rate. In total, the model detects 95% of the true correlations at 10% of false positives. We want to additionally point out that some of correlations present are very subtle; 13% of the correlations are evidenced by a single output packet. We also point out that CT-NOR performs significantly better than both alternatives based on co-occurrence of input packets, providing even more conclusive evidence that CT-NOR is capturing nontrivial dependencies. The approximation error from using the bound of section 5.3 is minimal, while the computation savings are significant. On a relatively slow laptop, the bounds on log-likelihood ratio test for a hour of traffic on a busy HTTP proxy can be computed in 7 seconds; exact computations take 86 seconds.

6.2 Changepoint Detection

Since the true presence or absence of a changepoint is unknown, we estimate it from the actual packet causes, obtained through deep inspection of HTTP packets. We collect a set of input and output channel pairs for which there is no evidence of change. We regard these as coming from the null hypothesis. A set of pairs for which the ground truth provides strong evidence of a change are collected, and considered to be from the alternative hypothesis.

We apply our changepoint test to that population, and report the results in Figure 4. The CT-NOR changepoint detection algorithm produces uniformly distributed p-values for channels which come from the null hypothesis and do not exhibit a changepoint, confirming that our null hypothesis distribution is calibrated. On the other hand, the test on alternative hypothesis channels produces a large proportion of very small p-values, indicating confidence that a changepoint occurred.

[5]As sometimes an input package generates more than one output packet, we enabled our model to account for this by allowing "autocorrelations" to take place. Namely a packet in an output channel can depend on an input channel or on the (time-wise) preceding output packet.

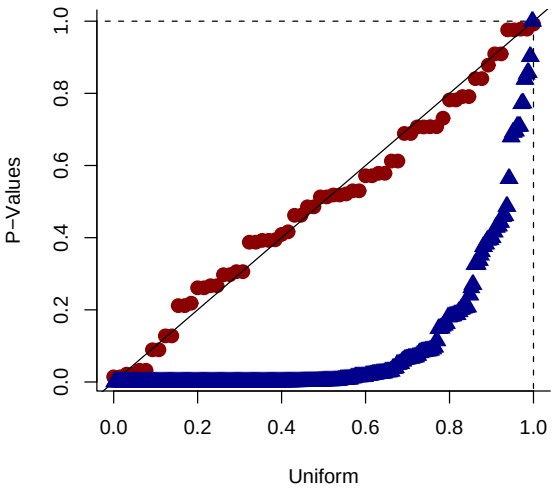


Figure 4: Quantile-Quantile plot of changepoint p-values. The red circles are channel pairs which, according to the ground truth, not exhibit a changepoint. The blue triangles represent channel pairs exhibiting change according to the ground truth.

7 Conclusions and Future Work

We presented a generative model based on Poisson processes called CT-NOR, to model the relationship between events based on the time of their occurrences. The model is induced from data only containing information about the time stamps for the events. This capability is crucial in the domain of networked systems as collecting any other type of information would entail prohibitive amounts of overhead. Specific domain knowledge about the expected shape of the distribution of the time delay between events can be incorporated to the model using a parameterized function. The EM algorithm used to fit the parameters of the model given the data also induces the parameters of this function. The combination of knowledge engineering and learning from data is clearly exemplified in the application we presented to the domain of computer systems, where we used a mixture model consisting of an exponential and a uniform distribution.

In terms of applying the model we focused on providing building blocks for diagnosis and monitoring.

We provided algorithms based on statistical hypothesis testing for (a) discovering the dependencies between input and output channels in computer networks, and for (b) finding changes in expected behavior (change-point detection). We validated these algorithms first on synthetic data, and then on a subset (HTTP traffic) of a trace of real data from events in a corporate communication network containing 500 computers and servers.

The relationship presented in Section 4 between CT-NOR and the NOR gate is interesting for multiple reasons. First, as the NOR gate has been extensively studied in this community in modeling and learning environment and in causal discovery [4], the immediate benefits are a) increasing the applicability to continuous time, and b) augmenting its modeling capabilities using the time delay functions used in this work. Second, this correspondence provide us with another intuition on the independence assumptions behind the Poisson process, as applied to the characterization of the relationship between the events in various inputs to the events in a specific output.

For the particular application of dependency discovery between channels in a computer network we explored a varied set of alternative approaches. They all failed miserably. Among these, we briefly discuss two: We cast the problem as one of classification, and tried a host of Bayesian network classifiers [6]. The idea was to first discretize time into suitable periods, and then have as features the existence or absence of events in the input channels and as the class the existence or absence of events in the output channel. The accuracy was abysmal. The main problem with this approach is that the communication in these networks is bursty by nature with relatively large periods of quiet time. Once we started to look at Poisson as the appropriate way to quantify the distributions in these classifiers the choice of the Poisson process became clear. We also explored the use of hypothesis testing comparing the inter-time between events in the input and output channels to the inter-time between the input and a fictitious random channel. The accuracy in terms of false positives and true positives was worse than those based on co-occurrence. The main problem here is that we are considering pairwise interactions and there are many confounder in all the other channels.

With regards to related approaches, both the work on continuous time Bayesian networks [10] and in general about dynamic Bayesian networks (e.g., [9]) are obviously very different in terms of the parameterization of the models, the assumptions, and the intended application. The work that is closest to ours is contained in the paper by Rajaram et al [12] where they propose a (graphical) model for point processes in terms of Poisson Networks. The main difference between their work and ours is the tradeoff between representation capabilities and complexity in inference that the different foci of our respective papers entails. Due to the distributed nature of our application domain, we concentrate on modeling the "families" (local parent/child relationship) and basically assume that we can reconstruct, in a distributed manner based on the local information, the topology of the network. This enables us to induce families with large numbers of parents, and with relatively complex interactions as given by the delay function f_θ, while performing inference efficiently. In the Poisson Networks paper [12], the number of parents of each node are restricted, and the rate function is parameterized by a generalized linear model. Even with these (relatively benign) restrictions inference is non-trivial in terms of finding the structure of the Bayesian network and indeed this is a contribution of that paper. Obviously, future work includes merging both approaches: an immediate benefit would be to decrease the vulnerability of our approach to spurious causal dependencies due to ignoring the global structure in the estimation.

There are other three threads that we are currently investigating for future work. The first one involves recasting the fitting and inference procedures described in the model in the Bayesian framework. An advantage of the Bayesian approach will be on the inclusion of priors. As channels differ greatly on the number of events this can further increase the accuracy of discovery. A second direction is that of incorporating False Discovery Rates [3] calculations in order to accurately estimate false positives when we don't have ground truth regarding the relationship between the channels. As we are performing a large number of hypothesis tests, this becomes a necessity. In [2] we experimented with the basic approach described in [3], and we verified that the approach is very conservative in the context of the HTTP and DNS protocols where we do have ground truth. We plan to explore less conservative approaches such as the one described in [13] or adapt the one explored in [8]. Finally we are in the process of getting suitable data and plan to apply this model to biological networks such as neurons that communicate with other neurons using spikes in electrical potential.

8 Acknowledgments

We thank T. Graepel for comments on a previous version of this paper. We are also grateful for the helpful suggestions of the anonymous reviewers which we hope we have addressed to their satisfaction.

References

[1] V. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *SIGCOMM'07*, Aug. 2007.

[2] P. Barham, R. Black, M. Goldszmidt, R. Isaacs, J. MacCormick, R. Mortier, and A. Simma. Constellation: automated discovery of service and host dependencies in networked systems. Technical Report MSR-TR-2008-67, Microsoft Research, 2008.

[3] Y. Benhamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society*, 57(2-3):125–133, 1995.

[4] J. Breese and D. Heckerman. Causal independence for probability assessment and inference in Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 1996.

[5] R. Durrett. *Probability: Theory and Examples*. Duxbury Press, second edition, 1996.

[6] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

[7] B. Lindsay. *Mixture Models: Theory, Geometry, and Applications*. Institute of Mathematical Statistics, Hayward, 1995.

[8] J. Listgarten and D. Heckerman. Determining the number of non-spurious arcs in a learned DAG model: Investigation of a Bayesian and a frequentist approach. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 2007.

[9] K. Murphy. Dynamic Bayesian networks: Representation, inference, and learning. Technical report, UC Berkeley, 2001. PhD Thesis.

[10] U. Nodelman, C. Shelton, and D. Koller. Learning continuous time Bayesian networks. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 2003.

[11] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[12] S. Rajaram, T. Graepel, and R. Herbrich. Poisson-networks: A model for structured point processes. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, 2005.

[13] J. D. Storey. A direct approach to false discovery rates. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(3):479–498, 2002.

[14] L. Wasserman. *All of Statistics*. Springer, 2004.

Efficient inference in persistent Dynamic Bayesian Networks

Tomáš Šingliar
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15213

Denver H. Dash
Intel Research and
Department of Biomedical Informatics
University of Pittsburgh
Pittsburgh, PA 15213

Abstract

Numerous temporal inference tasks such as fault monitoring and anomaly detection exhibit a *persistence* property: for example, if something breaks, it stays broken until an intervention. When modeled as a Dynamic Bayesian Network, persistence adds dependencies between adjacent time slices, often making exact inference over time intractable using standard inference algorithms. However, we show that persistence implies a regular structure that can be exploited for efficient inference. We present three successively more general classes of models: *persistent causal chains* (PCCs), *persistent causal trees* (PCTs) and *persistent polytrees* (PPTs), and the corresponding exact inference algorithms that exploit persistence. We show that analytic asymptotic bounds for our algorithms compare favorably to junction tree inference; and we demonstrate empirically that we can perform exact smoothing on the order of 100 times faster than the approximate Boyen-Koller method on randomly generated instances of persistent tree models. We also show how to handle non-persistent variables and how persistence can be exploited effectively for approximate filtering.

1 Introduction

Persistence is a common trait of many real-world systems. It is used to model permanent changes in state, such as when components of a system that have broken until someone intervenes to fix them. Especially interesting and useful are diagnostic models where misalignments and other process drifts may cause a cascade of other failures, all of which may also persist until the root cause is fixed. Even when such changes are not truly permanent, they are often reversed slowly relative to the time scale of the model, and persistence can be a good approximation in such systems. For instance, vehicular accidents cause obstructions on the road that last much longer than the required detection time and are thus persistent for the purpose of detection [20]. Another example is outbreak detection [4], where an infected population stays infected much longer than the desired detection time. There are many other examples of persistence and approximate persistence.

Dynamic Bayesian Networks (DBNs) [5] are a general formalism for modeling temporal systems under uncertainty. Many standard time-series methods are special cases of DBNs, including Hidden Markov Models [18] and Kalman filters [7]. Discrete DBNs in particular are a very popular formalism, but usually suffer from intractability [1] when dense inter-temporal dependencies are present among hidden state variables, leading many to search for approximation algorithms [1, 13, 15, 14]. Unfortunately, modeling persistence with DBNs requires the introduction of many inter-temporal arcs, often making exact inference intractable with standard inference algorithms.

In this paper, we define Persistent Causal DBNs (PC-DBNs), a particular class of DBN models capable of modeling many real-world systems that involve long chains of causal influence coupled with persistence of causal effects. We show that a linear time algorithm exists for inference (smoothing) in linear chain and tree-based PC-DBNs. We then generalize our results to polytree causal networks, where the algorithm remains exact, and to general networks, where it inherits its properties of loopy belief propagation [21]. Our method relies on a transformation of the original prototype network, allowing smoothing to be done efficiently; however, this method does not readily deal with the incremental filtering problem. Nonetheless, we show empirically that, if evidence is observed at every time slice, approximate filtering can be accomplished with fixed window smoothing, producing lower error than approximate Boyen-Koller (BK) filtering [1]

using a fraction of the computation time.

The algorithm that we present exploits a particular type of determinism that is given by the persistence relation. There has been other work that seeks to directly or indirectly exploit general deterministic structure in Bayesian networks using compilation approaches [2], a generalized version belief propagation [10], and variable elimination with algebraic decision diagrams [3, 19]. These more general methods have not been tailored to the important special cases of DBNs and persistency. To our knowledge, this is the first work to investigate persistency in DBNs.

The paper is organized as follows: In Section 2 we introduce the changepoint transformation. Section 3 introduces persistent causal chain DBNs and the corresponding inference algorithm, which retains all the essential properties of later models. Then, Section 4 will discuss the steps leading to a fully general algorithm. Experimental results are presented in Section 5, followed by conclusions.

2 Notation and changepoints

Consider a Bayesian network (BN) with N binary variables X_i; we will refer to this network as the *prototype*. The corresponding Dynamic BN with M slices is created by replicating the prototype M times and connecting some of the variables to their copies in the next slice. In our notation, upper indices range over time slices of the DBN; lower indices range over variables in each time slice. Colon notation is used to denote sets and sequences. Thus, for instance, $X_4^{1:M}$ denotes the entire temporal sequence of values of X_4 from time 1 to time M. Variables without an upper index will refer to their respective counterparts in the prototype. We say that a variable X_k is *persistent* if

$$P(X_k^t = 1 | X_k^{t-1}, U^t) = \begin{cases} P(X_k|U) & \text{if } X_k^{t-1} = 0 \\ 1 & \text{if } X_k^{t-1} = 1 \end{cases}, \quad (1)$$

where $U = Pa(X_k)$ refers to the parents of X_k in the prototype. In other words, 1 is an absorbing state. Sometimes [12] a variable is called persistent if it has an arc to the next-slice copy of itself. Our definition of *persistence* is strictly stronger, but no confusion should arise in this paper.

There are 2^M temporal sequences of values of a binary variable X_k. If the variable is persistent, the number of configurations is reduced to $M+1$. Information about $X_k^{1:M}$ can be summarized by looking at the time when X changed from 0 to 1 (we sometimes refer to the 0 state as the *off* state and 1 as the *on* state). Thus, inference in the persistent DBN with binary variables is equivalent to inference in a network whose topology closely resembles that of the prototype and whose variables are $M+1$-ary discrete *changepoint* variables, with correspondingly defined conditional probability distributions (CPDs), as shown in Figure 1b. The models in Figure 1a and 1b are identical; one can go back and forth between them by recognizing that

$$\begin{aligned} (\tilde{X} = j) &\Leftrightarrow (X^j = 0) \wedge (X^{j+1} = 1) \text{ and} \\ (X^j = 0) &\Leftrightarrow (\tilde{X} > j). \end{aligned}$$

If the prototype is a tree, belief propagation in the transformed network yields an algorithm whose complexity is $O(M^2N)$. The quadratic part of the computation comes from summing over the $M+1$ values of the single parent for each of the $M+1$ values of the child. Similarly, if the prototype is a polytree, complexity will be proportional to $M^{U_{max}+1}$, where U_{max} is the largest in-degree in the network. This transformation by itself, when all hidden state variables are persistent, allows us to perform smoothing much more efficiently than by operating on the original DBN. There is, however, additional structure in the CPDs that allows us to do better by a factor of M, and we can also adapt our algorithm to deal with the case when some hidden variables are not persistent.

3 PCC-DBN inference

To simplify the exposition, let us now focus on a specific prototype, a persistent causal chain DBN (PCC-DBN). This is a chain with $Pa(X_i) = \{X_{i-1}\}, i = 1, ..., N$ and $Pa(O) = X_N$ (thus it has N+1 nodes). Let us further assume that the leaves are non-persistent and observed, while the causes (X nodes) are all persistent and hidden. The network is shown in Figure 1a and its transformed version in Figure 1b.

Consider the problem of computing $P(O)$. This is in general one of the most difficult inference problems, requiring one to integrate out all hidden state variables, and is implicit in most inference queries:

$$P(O^{1:M}) = \sum_{X_{1:N}^{1:M}} P(O^{1:M} \mid X_{1:N}^{1:M}) \cdot P(X_{1:N}^{1:M}) \quad (2)$$

Let $\{j_k : 0 \leq j_k \leq M\}$ index the sequence of $X_k^{1:M}$ in which variable $X_k^{j_k}$ is the last (highest-time) variable to be in the *off* state, unless $j_k = 0$ in which case it indexes the sequence in which all X_k are in the *on* state. As an example, if $M = 3$, then $j_k = \{0, 1, 2, 3\}$ indexes the states $X_k^{1:M} = \{111, 011, 001, 000\}$, respectively, for all k. All configurations not indexed by j_i have zero probability due to the persistence assumption. To simplify notation, we use j_k to denote the event that $X_k^{1:M}$ is the sequence indexed by j_k. We also say that X_k *fired* at j_k. We can decompose Equa-

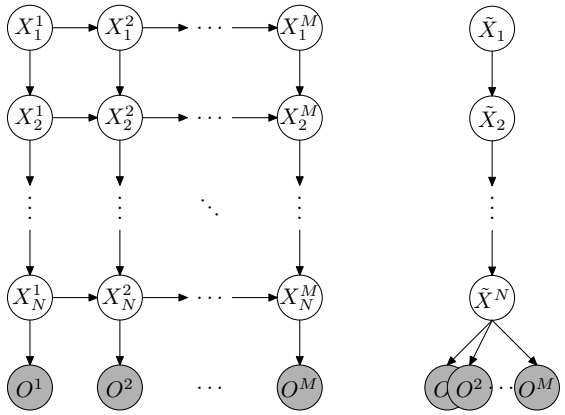


Figure 1: **(a)** A PCC-DBN network with $N+1$ nodes per slice; **(b)** the transformed network. We sometimes refer to X_2^1 as the temporal parent of X_2^2 and to X_1^2 as its causal parent.

tion 2 according to the network structure as follows:

$$P(O^{1:M}) = \sum_{j_1=0}^{M} P(j_1) \sum_{j_2=0}^{M} P(j_2 \mid j_1) \dots \\ \dots \sum_{j_N=0}^{M} P(j_N \mid j_{N-1}) \cdot P(O^{1:M} \mid j_N) \quad (3)$$

Denote by P_k the probability that variable X_k will fire for the first time given that its causal parent *has* fired, and by $\widehat{P}_k$ the probability that X_k will fire for the first time given that its causal parent has not fired:

$$\begin{aligned} P_k &\equiv P(X_k^j = 1 \mid X_k^{j-1} = 0, X_{k-1}^j = 1) \\ \widehat{P}_k &\equiv P(X_k^j = 1 \mid X_k^{j-1} = 0, X_{k-1}^j = 0). \end{aligned}$$

Let $P_{\overline{k}}$ and $\widehat{P}_{\overline{k}}$ denote the complements $1 - P_k$ and $1 - \widehat{P}_k$, respectively. We can define Σ_k^L recursively to denote the partial sum over j_k from Equation 3, conditioned on $j_{k-1} = L$:

$$\Sigma_k^L \equiv \sum_{j_k} P(j_k \mid j_{k-1} = L) \cdot \Sigma_{k+1}^{j_k} \quad (4)$$

with boundary condition $\Sigma_{N+1}^L \equiv P(O^{1:M} \mid j_N = L)$. Using this notation, Equation 3 can be rewritten as:

$$P(O^{1:M}) = \sum_{j_1=0}^{M} P(j_1) \cdot \Sigma_2^{j_1} \quad (5)$$

Now we now need to show that one can calculate the entire set $\Sigma_{2:N}^{0:M}$ in time $O(MN)$. Each Σ_k^L can be written as follows:

$$\Sigma_k^L = \bar{\sigma}_k^L + \sigma_k^L + \hat{\sigma}_k^L, \quad (6)$$

where $\bar{\sigma}_k^L$ contains all the terms in the sum such that X_k first fires when X_{k-1} has not fired:

$$\bar{\sigma}_k^L = \sum_{j_k < L} \widehat{P}_{\overline{k}}^{j_k} \widehat{P}_k \cdot \Sigma_{k+1}^{j_k}. \quad (7)$$

σ_k^L contains all the terms in which X_k first fires when X_{k-1} has *also* fired:

$$\sigma_k^L = \sum_{L \le j_k < M} \widehat{P}_{\overline{k}}^{L} P_{\overline{k}}^{j_k - L} P_k \cdot \Sigma_{k+1}^{j_k}, \quad (8)$$

and $\hat{\sigma}_k^L$ contains the final term in which X_k never fires:

$$\hat{\sigma}_k^L = \widehat{P}_{\overline{k}}^{L} P_{\overline{k}}^{M-L} \cdot \Sigma_{k+1}^{M}. \quad (9)$$

In order to calculate Equation 5 in time $O(MN)$, we need to pre-compute $\bar{\sigma}_k^L$, σ_k^L and $\hat{\sigma}_k^L$ for all values of L in $O(M)$ for each variable X_k.

3.1 Upward Recursion Relations

As a boundary condition for the recursion, assume we have calculated Σ_{N+1}^k for all $0 \le k \le M$. We show how to do this in time $O(M)$ in Section 3.2. Also, this algorithm requires the pre-calculation and caching of $\widehat{P}_{\overline{k}}^i$ for $0 \le k \le N$ and $0 \le i \le M$, which can be done recursively in $O(MN)$ time and space.

Inspecting Equation 7 more closely, it should be easy to see that one can calculate $\bar{\sigma}_N^i$ for $0 \le i \le M$ in $O(M)$ time using the following recursion:

$$\bar{\sigma}_l^{i+1} = \bar{\sigma}_l^i + \widehat{P}_{\overline{l}}^i \cdot \widehat{P}_l \cdot \Sigma_{k+1}^i, \quad (10)$$

with boundary condition $\bar{\sigma}_l^0 = 0$ for all l. One can also calculate σ_k^i for $0 \le i \le M$ with the recursion:

$$\sigma_k^{i-1} = \frac{\sigma_k^i}{\widehat{P}_{\overline{k}}} P_{\overline{k}} + \widehat{P}_{\overline{k}}^{i-1} \cdot P_k \cdot \Sigma_{k+1}^{i-1}, \quad (11)$$

with boundary condition $\sigma_k^M = 0$ for all l. Finally, one can calculate $\hat{\sigma}_N^i$ for $0 \le i \le M$ with the recursion:

$$\hat{\sigma}_k^{i-1} = \frac{\hat{\sigma}_k^i}{\widehat{P}_{\overline{k}}} P_{\overline{k}} \quad (12)$$

with boundary condition $\hat{\sigma}_k^M = \widehat{P}_{\overline{k}}^M \cdot \Sigma_{k+1}^M$ for all l.

Once $\bar{\sigma}_N^i$, σ_N^i and $\hat{\sigma}_N^i$ are calculated, one can calculate all Σ_N^i for $0 \le i \le M$ in $O(M)$ time using Equations 10, 11, 12 and 6. After $\Sigma_N^{0:M}$ is calculated, we can use Equation 4 to obtain $\Sigma_{N-1}^{0:M}$ in time $O(M)$, and repeat N times to get all values of $\Sigma_{1:N}^{0:M}$. Thus the entire calculation takes $O(MN)$ time.

3.2 Computing Σ^i_{N+1}

To finalize the proof, we have to show how to calculate Σ^i_{N+1} (the probability of the observations for a given configuration i of $X_N^{1:M}$) for *all* $0 \leq i \leq M$ in time $O(M)$. Recall that $\Sigma^i_{N+1} \equiv P(O^{1:M} \mid j_N = i)$. Since the parent of each O^j is given, for each i, this calculation is simply the product of the observations:

$$P(O^{1:M} \mid j_N = i) = \prod_{k=1}^{M} P(O^k \mid X_N^k, j_N = i) \quad (13)$$

Using our existing notation, we define

$$\phi^\ell_{N+1} = P(O^k \mid X_N^k = 1), \quad (14)$$
$$\bar{\phi}^\ell_{N+1} = P(O^k \mid X_N^k = 0), \quad (15)$$

Σ^i_{N+1} can be calculated for all $0 \leq i \leq M$ in time $O(M)$ via the recursion relation:

$$\Sigma^0_{N+1} = \prod_{\ell=1}^{M} \phi^\ell_{N+1} \quad \text{and} \quad \Sigma^{\ell+1}_{N+1} = \Sigma^\ell_{N+1} \cdot \frac{\bar{\phi}^\ell_{N+1}}{\phi^\ell_{N+1}}. \quad (16)$$

Note that this formulation puts no distributional assumption on $P(O|X_N)$. The leaves can be distributed as multinomials, Gaussians etc, as is often done with Hidden Markov models [18] when they are put to their many uses.

3.3 Downward Recurrences

The above discussion completes the description of the "λ-pass" of PCC-DBN algorithm. Similar reasoning can be applied to obtain the "π-pass" recurrences that we now give without full derivation. Analogously to Σ, the semantics of Ψ^j_k is $p(X_k = j|O^+_k)$, where O^+_k is the subset of evidence reachable from X_k through its parent[1]. Ψ^j_k is again a sum of three components:

$$\Psi^j_k = \psi^j_k + \bar{\psi}^j_k + \hat{\psi}^j_k \quad (17)$$

$\bar{\psi}$ accounts for the terms where the parent has not yet changed:

$$\bar{\psi}^{\ell-1}_k = \bar{\psi}^\ell_k \cdot \frac{1}{\overline{P}_{\overline{k}}} + \widehat{P}^{\ell-1}_{\overline{k}} \cdot \widehat{P}_k \cdot \Psi^\ell_{k-1} \quad (18)$$

with initialization $\bar{\psi}^M_k = 0$ for all k.

ψ accounts for the terms where the parent has already changed:

$$\psi^{\ell+1}_k = \psi^\ell_k \cdot P_{\overline{k}} + \widehat{P}^{\ell+1}_{\overline{k}} \cdot P_k \cdot \Psi^{\ell+1}_{k-1} \quad (19)$$

[1]We only have evidence in the bottom layer in PCC-DBNs, but this will come handy in the next section.

with boundary condition $\psi^0_k = P_k \Psi^0_{k-1}$ for all k. Also, since X_k eventually changes in this scenario, $\psi^M_k = 0$.

$\hat{\psi}$ accounts for the terms where the node never changes:

$$\hat{\psi}^M_k = \sum_{0 \leq i \leq M} \widehat{P}^i_{\overline{k}} \cdot P^{M-i}_{\overline{k}} \cdot \Psi^i_{k-1}. \quad (20)$$

Because the upper index refers to the changepoint of X_k, only $\hat{\psi}^M_k$ is non-zero. We can just compute this in $O(M)$ without the need for recurrences.

Initialization of the Ψ-recurrences happens at the root(s) of the network. For any root r, $\Psi^0_r = \widehat{P}_k$ and recurrently $\Psi^{i+1}_r = \Psi^i_r \cdot \widehat{P}_{\overline{r}}$. Finally, $\Psi^M_r = \Psi^{M-1}_r \widehat{P}_{\overline{r}} / \widehat{P}_k$.

3.4 PCC-DBN and belief propagation

We have just defined PCC-DBN, a version of belief propagation that first collects the evidence by passing the λ-messages towards the root of the chain and the proceeds to distribute information towards the leaves via the π-messages. After propagation is complete, we can obtain any posterior as

$$p(\tilde{X}_k|O) = \Sigma_{k+1} \cdot \Psi_k. \quad (21)$$

It is now useful to recall the types of potentials involved in Pearl's algorithm [8] and how they relate to the quantities above. For each node X, there are local potentials $\pi_X(x) \equiv_{def} p(X = x|e^+_X)$ and $\lambda_X(x) \equiv_{def} p(e^-_X|X = x)$, where e^+_X and e^-_X denote respectively the evidence reachable through parents and the evidence reachable from X "downwards", X included. There are two types of messages in Pearl's algorithm: $\pi_{X \to Y_i}$ sent by X to its children and $\lambda_{X \to U_i}$ sent to its parents. A closer look at PCC-DBN reveals that each Σ_k is identical to $\lambda_{X_k \to X_{k-1}}$ — the message from X_k to its single parent X_{k-1}. The local potential $\lambda_{X_k}(j_k)$ is identical to Σ_{k+1}, because there are no children other than X_{k+1} and evidence is only observed at the bottom of the chain. Ψ_k corresponds directly to $\pi_{X_k}(j_k)$. This is why Equation 21 works.

3.5 Simple Generalizations and Causal Trees

While PCC-DBNs are useful for demonstrating the general ideas of handling the probability distributions arising from the changepoint transformation, they form a rather restricted class of networks, and the inference query that we performed was also restricted. Here we state succinctly a set of simple alterations which allow this algorithm to be relaxed in various ways:

General evidence patterns We can have observations anywhere in the network, in any time slice.

Casting the inference as belief propagation gives the answer to any probabilistic query as Equation 21 with one caveat: An observation such as $X_3^3 = 1$ does not tell us with certainty the position of the changepoint, but it just provides evidence that $j_3 < 3$, thus we cannot simply set the changepoint variable to state 3. Rather, the potentials corresponding to such evidence must be multiplied onto the messages as prescribed by the belief propagation algorithm (see Equation 22).

Non-stationarity Stationarity of conditional probability distributions was used to simplify the formulae in the previous exposition, but is not required. All that is needed is to keep running products of respective probabilities instead of the powers in the exponents of P_k, $\widehat{P}_k$. They need to be computed incrementally and tabulated to avoid hidden linear terms in the computation.

Extension to trees The extension of PCC-DBN to causal trees (PCT-DBNs) is now fairly straightforward. Because each node X_k can now have multiple children $Ch(X_k)$, we must replace Σ_k in all recurrences with the true λ-potential for X_k:

$$\lambda_{X_k} = \lambda_{X_k \to X_k} \cdot \prod_{i \in Ch(X_k)} \Sigma_i, \tag{22}$$

where $\lambda_{X_k \to X_k}$ accounts for evidence observed in X_k's temporal chain. The vector $\lambda_{X_k \to X_k}$ is zero where the evidence rules out a changepoint — before the time t of the last observed $X_k^t = 0$ and after the time s of first observed $X_k^s = 1$. Everywhere else, $\lambda_{X_k \to X_k}(j_k) = 1$. Note that $\lambda_X(x)$ potential can be obtained in $O(M)$ time per node.

In computation of ψ potentials, Ψ_k on the right-hand side of the recurrences is replaced by the $\pi_{Pa(X_k) \to X_k}$, which in turn include the influence of evidence under X_k's siblings:

$$\pi_{Pa(X_k) \to X_k} = \pi_{X_k} \cdot \prod_{i \in Ch(Pa(X_k)) \setminus X_k} \Sigma_i. \tag{23}$$

Again, this preserves the $O(M)$ per-node complexity. Thus, PCT-DBN is linear in both N and M.

4 Further Generalizations

In this section we describe three more important generalizations of PC-DBNs: polytrees, non-persistent nodes, and finally an approximate algorithm for general DAGs. These relaxations are more involved than those of Section 3.5 and thus require more elaboration.

4.1 Polytrees

Belief propagation [16, 17] is a powerful framework for exact inference in polytree networks. Polytrees, unlike trees, allow multiple parents of a node, but remain acyclic in the undirected sense. In polytree changepoint networks, structure in the conditional probability table $P(X = x|U)$ can be exploited to save a multiplicative factor of $M + 1$ just as we showed for tree networks. The ψ-recurrences run over the first parent variable, while the remaining parents are summed over by brute force. Similarly, the σ-recurrences run over the parent that the message is addressed to. For instance, the definition of σ will be replaced by

$$\sigma_k^i \propto \sum_{L \le j_k < M} \left[\prod_{z=1}^{L} \widehat{P}_{\overline{k}}^{(z)} \right] \cdot \left[\prod_{z=L+1}^{j_k} P_{\overline{k}}^{(z)} \right] P_k^{(j_k+1)} \cdot \Sigma_{k+1}^{j_k} \tag{24}$$

and Equation 11 by

$$\begin{aligned} \sigma_k^{i-1} &= \sigma_k^i \cdot \frac{P_{\overline{k}}^{(i)}}{\widehat{P}_{\overline{k}}^{(i)}} + \left[\prod_{z=1}^{i-1} \widehat{P}_{\overline{k}}^{(z)} \right] \cdot P_k^{(i)} \cdot \Sigma_{k+1}^{i-1} \\ \sigma_k^M &= 0, \end{aligned} \tag{25}$$

where, assuming we are sending to the first parent,

$$\begin{aligned} P_k^{(z)} &= P(X_k = 1 | U_1 = 1, I\{\mathbf{U}_{2:} \le z\}) \\ \widehat{P}_k^{(z)} &= P(X_k = 1 | U_1 = 0, I\{\mathbf{U}_{2:} \le z\}) \end{aligned} \tag{26}$$

are now functions of the joint configuration of the remaining parents $\mathbf{U}_{2:}$. The proportionality constant in Equation 24 equals the product of the remaining parents' π-messages. We call this PPT-DBN, the persistent polytree algorithm.

The worst-case time complexity of PPT-DBN is dominated by the cost associated with the largest family-clique: $O((M+1)^{U_{max}})$. The $2TBN$ algorithm [12] suffers a worst-case time complexity $O(2^{2N}M)$, as all nodes in two slices may be entangled [9] in the clique to connect the two subsequent time-slices, even though the prototype network is a polytree [12, section 3.6.2]. Therefore, we expect PPT-DBN will be comparatively better for shorter temporal chains of larger networks. However, PPT-DBN really shines on space complexity. At most $O(MN)$ memory is consumed, compared to 2TBN, where the potentials in the joint tree can grow as large as $O(2^{2N})$. Later we show experimentally how dramatic the difference can be.

4.2 Non-persistent nodes

While it is convenient to assume that all non-leaf variables are persistent, it does limit the modeling power at our disposal. We now show how an occasional non-persistent variable in the network can also be handled in polynomial time. We assume the non-persistent variable is isolated, that is, all of its neighbors are persistent. We make this assumption in order to avoid having to invoke an embedded general DBN inference

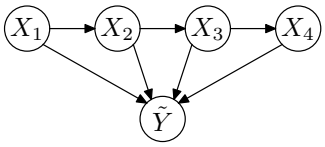


Figure 2: The minimal example of network with a non-persistent node.

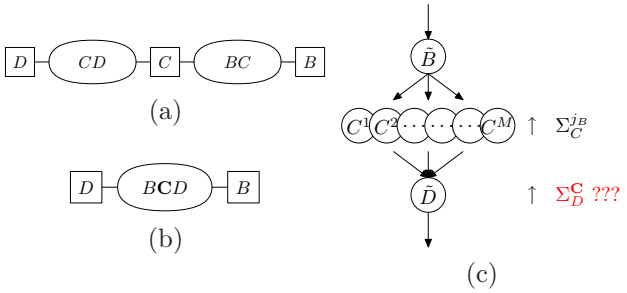


Figure 3: **a)** Induced cliques and separators **b)** Enlarged clique for generalized BP **c)** Σ-message flow in a network with a non-persistent variable

algorithm such as 2TBN to handle connected non-persistent variables. It can be done, but quickly becomes complex and inelegant. A simple way to handle connected non-persistent nodes is to combine them into a single joint node. Obviously, this solution causes exponential growth in the state space of the joined nodes, making it somewhat unappealing. A two-slice approach made aware of the determinism, e.g. by use of ADD compilation [3, 2], could very well work better for networks with only a few persistent variables.

4.2.1 A simple example

To illustrate how an isolated non-persistent node would be handled, assume first a simple structure such as in Figure 2. Then we can efficiently compute $P(\tilde{Y}=j)$ by moving the sums inward:

$$P(\tilde{Y}=j) = \sum_{\mathbf{X}} P(\mathbf{X})P(\tilde{Y}=j|\mathbf{X}) =$$
$$\sum_{X^1,\ldots X^M} \left[\prod_{k=1}^{M} P(X^k|X^{k-1})\right]\left[\prod_{k=1}^{M}\phi_j^k\right] =$$
$$\underbrace{\sum_{X^1} P(X^1)\phi_j^1 \underbrace{\sum_{X^2} P(X^2|X^1)\phi_j^2 \ldots \underbrace{\sum_{X^M} P(X^M|X^{M-1})\phi_j^M}_{\kappa^M}}_{\kappa^2}}$$

This gives rise to the recurrence

$$\kappa_i^M(j) = \sum_v P(X^M=v|X^{M-1}=i)\cdot\phi_j^M \qquad (27)$$
$$\kappa_i^k(j) = \sum_v P(X^k=v|X^{k-1}=i)\cdot\phi_j^k\cdot\kappa_v^{k+1},$$

with ϕ_j^k defined appropriately:

$$\phi_j^k(X^k) = \begin{cases} P(Y^1=0|X^1) & \text{if } k=1 \\ P(Y^k=0|Y^{k-1}=0,X^k) & \text{if } 1<k\le j \\ P(Y^k=1|Y^{k-1}=0,X^k) & \text{if } k=j+1 \\ 1 & \text{if } k>j+1 \end{cases}$$

Now, $P(\tilde{Y}=j)=\kappa_1^0$. Moreover, a short analysis will reveal that

$$\kappa_i^k(j) = \kappa_i^{k+1}(j+1) \qquad \forall 1\le k<j<M-1.$$

Therefore, we do not need to compute κ for every j, but compute $\kappa_i^k(M)$ for all k as a special case and then $\kappa_i^k(M-1)$ for all k to start the recursion. All other values can be read off $\kappa_i^k(M-1)$ with the appropriate indexing shift. Thus, we can obtain the entire distribution $P(\tilde{Y})$ in $O(M)$ time! Allowing non-persistent variables to take on multiple values is also straightforward: we only need to allow the bottom index in κ_i^k to range over the domain of X_k.

4.2.2 The general case

Pearl's belief propagation has been generalized to the clique tree propagation algorithm [21]. With belief propagation (BP), the cliques correspond to edges of the original polytree and the separators consist of single nodes. In the process of message passing, the variables not in the separator are summed out of the clique potentials.

The PCT-DBN algorithm used the "natural" cliques induced by the transformed network. Assume we have a situation such as in Figure 3. Because the variable C is not persistent, the size of the induced separator $\boxed{\mathbf{C}}$ is 2^M. However, we can work conceptually with a larger clique $B\mathbf{C}D$. Message propagation then calls for summing out all $C^{1:M}$, which we can do without actually instantiating the clique potential using a recurrence derived much like that of Equation 27. In the interest of space, we only show here the simple κ recurrence. The full recurrence calls for summing over all persistent variables in the clique and the resulting complexity is $O((M+1)^B)$, where B is the number of persistent neighbors of the non-persistent variable.

5 Experimental evaluation

We implemented our algorithms in Matlab and compare them to the exact and approximate algorithms as implemented in the Bayesian Network Toolbox (BNT) [11]. Namely, we will compare to the Boyen-Koller (BK) algorithm [1] in its 1) exact and 2) fully factored setting. Although BK reduces in its exact form to the incremental junction-tree algorithm, we found it was faster in practice than the 2TBN implementation.

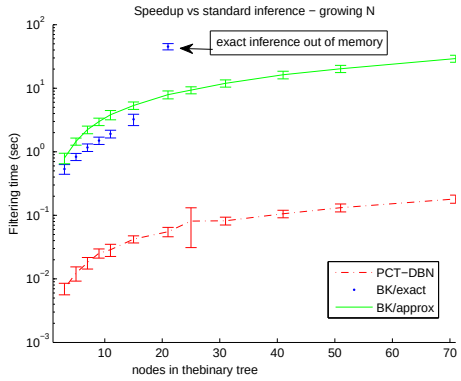


Figure 4: Performance scale up of PCT-DBN with N. The temporal length was fixed at $M = 20$. Note log scale y-axis.

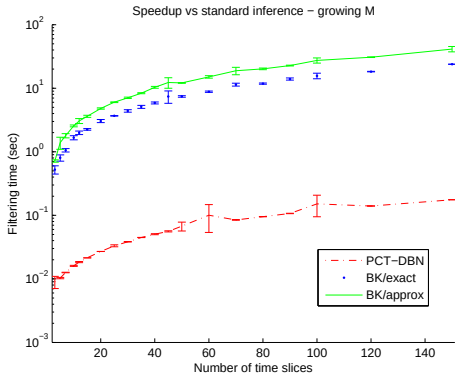


Figure 5: Performance scale up of PCT-DBN with M. The number of nodes was held at $N = 19$.

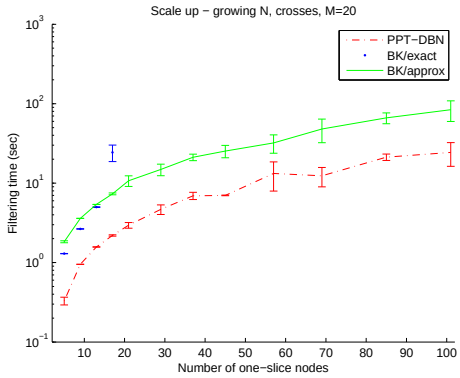


Figure 6: Performance scale up of PPT-DBN with N.

Therefore the 2TBN algorithm is not included in the evaluation.

Matlab run-time is not the ideal measure of algorithm complexity as it is arguably more sensititve to the quality of implementation compared to other languages. However, we should note that we did not make any special effort to optimize our code for Matlab, and the BNT library is a widely used and mature code base, so we expect any advantages due to code quality to fall to the competing approaches. Our Matlab code and further evaluation results can be downloaded at `http://www.cs.pitt.edu/~tomas/papers/UAI08`.

5.1 Speed of the tree algorithm

To compare inference speed, a network with the structure of a full binary tree with N nodes was generated. Among the MN possible observations, 10% of the variables were set to a random value (subject to persistence constraints so that $P(E) \neq 0$). We measured the time to execute the query $p(\tilde{X}_1|E)$—the posterior probability over the root node—for each algorithm[2]. This process was repeated 100 times for each M, N combination and the respective times added up. The results are graphed out in Figures 4 and 5.

PCT-DBN outperforms both the exact incremental joint tree algorithm and the approximate BK algorithm (assuming independence) by several orders of magnitude as N, the size of a slice grows (Figure 4). In fact, the exact algorithm soon runs out of memory (around $N = 20$) and only the approximate version keeps up. Exact PCT-DBN inference also performs consistently about 100 times faster than exact junction-tree and approximate BK inference when we look at scale-up with the number of slices, as shown in Figure 5.

[2]The actual query is in fact irrelevant as all algorithms compute all posterior marginals simultaneously.

5.2 Speed of the polytree algorithm

The asymptotic time complexity of PPT-DBN, as M increases, may be less favorable than that of the incremental approaches. However, its lower memory complexity is very favorable, as documented by the following experiment. We generated a network where most non-root nodes have exactly 2 parents and measured the time for the three inference algorithms. Quadratic scale-up with M is expected for PPT-DBN in such a network.

Figure 6 shows the *exact* PPT-DBN algorithm to be several times faster than, but scaling very similarly to, the *approximate* fully factorized Boyen-Koller algorithm with an $M = 20$ time slice inference window. Peeking ahead into Figure 7 suggests the time performance would be about identical at $M = 70$ time slices. The junction-tree algorithm does not scale beyond 20 nodes due to memory usage.

Figure 7 shows clearly that asymptotically, PPT-DBN

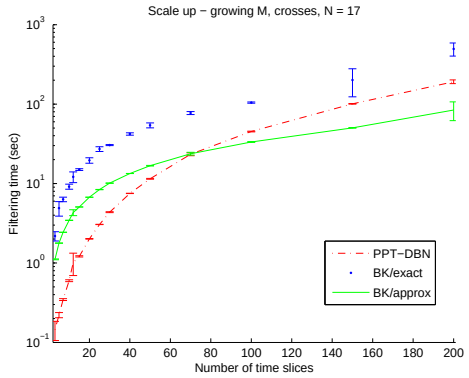


Figure 7: Performance scale up of PPT-DBN with M. The number of nodes was held at $N = 17$.

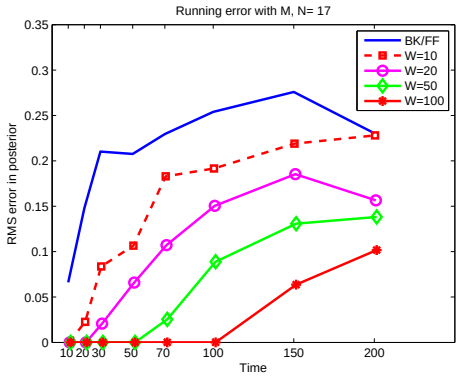


Figure 8: Accuracy of PC-DBN with growing inference window M. Averages are over 100 different parameterizations of the network; error bars are omitted for clarity but standard deviations are the same order of magnitude as the means for all curves.

scales with a steeper slope than both BK inference and junction-tree inference. Indeed, for about $N = 70$, BK eventually surpasses PPT-DBN in terms of speed. However, it remains faster than junction-tree incremental inference throughout the range. On a computer with 1 GB RAM, the exact version begins to hit memory limits around $M = 200$ and $N = 17$.

We conclude that if exact inference is desired for persistent polytree causal networks, using the PPT-DBN algorithm is a better choice for a wide range of inference window lengths. Furthermore, if approximate inference is acceptable, we show in Section 5.3 that for large enough N, fixed window smoothing using PPT-DBN can outperform BK inference in terms of RMS error, while still performing many times faster. For the special case of persistent causal trees, the new algorithm dominates by orders of magnitude in all ranges that we tested versus both junction tree and BK assuming intra-slice independence.

5.3 Fixed-window approximation

A minor disadvantage of the PPT-DBN algorithm is that it cannot do online inference yet. Therefore, when monitoring a process, M grows and so does the computation time. In practice, only a fixed number of most recent observations are usually considered with older observations falling out of the "window". Thus we evaluate if reasonable precision can be attained with small window sizes, where PPT-DBN dominates.

Figure 8 shows, for several time slices t, the root mean square error of computed posterior marginals

$$Err^t_{BK} = \sqrt{\sum_{i=1}^{N} [P_{BK}(X^t_i|O^{1:t}) - P_{ex}(X^t_i|O^{1:t})]^2}$$

incurred by the fully factored Boyen-Koller method and the same error for PPT-DBN which ignores all evidence older than W, for different values of W. We use a binary tree prototype with all leaf variables non-persistent and observed. All non-leaf variables are persistent and hidden. The CPT probabilities are sampled uniformly at random. The observed evidence O is obtained by forward-sampling the DBN and restricting it to the observables. We find that the error of our algorithm falls with growing W as expected. The results become even more favorable for PC-DBN as N, the number of nodes per slice, grows (see also further results online). The error made by fixing the inference window tends to be lower than that of the Boyen-Koller approximation for reasonable values of W and we can eliminate the unfavorable dependence on M at a small price of accuracy. One clear drawback to a naive implementation of the fixed-window approach is that if evidence is not observed at each time slice, in the presence of persistence a piece of crucial evidence might drop off the window preventing the model from "remembering" that a persistent state was already acheived. This glitch could in principle be fixed by caching when persistent variables have turned on.

6 Conclusions and future work

We presented an algorithm for PC-DBNs, a way to exploit the special structure of the DBN probability distribution when many variables are persistent. Unlike forward-backward approaches to DBN inference that work slice-to-slice, we collapse the entire temporal progression and perform inference in the original prototype network structure. For trees, the algorithm is many times faster than state-of-the-art general-purpose exact *and approximate* DBN inference algorithms, while having a space complexity of only

$O(MN)$. This continues to hold even in the polytree generalization with inference window lengths into the hundreds. While this method does not directly yield an incremental filtering algorithm, we show that a fixed-window smoothing version of PC-DBN inference can perform approximate filtering faster and with comparable or less error than BK-filtering.

Although we have not presented a filtering algorithm that can exploit persistence, we do believe that one is possible. The number of possible joint configurations of variables in two subsequent slices is 3^N with the persistence assumption as opposed to 4^N in the general network. This hints at the possibility of a 2TBN-like algorithm leveraging persistence and still remaining linear in the number of time slices.

Another possible direction for this work is to allow multi resolution temporal modeling by modeling systems on very short time scales, but utilizing a persistence approximation for the slow processes. In such cases, a model with a single time-scale could efficiently and accurately deal with systems that have both fast and slow processes.

Also interesting is the vision of approximate inference algorithms not requiring persistence, but simply assuming that the hidden state changes at most once in the period of interest. If the change in the hidden state is relatively slow, this could be a fairly accurate approximation. Such problems are often found in bioinformatics areas such as phylogeny discovery, where time of a mutation is of interest [6].

References

[1] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proceedings of UAI-98*, pages 33–42, 1998.

[2] Mark Chavira and Adnan Darwiche. Compiling Bayesian networks with local stucture. In *Proceedings of IJCAI 2005*, 2005.

[3] Mark Chavira and Adnan Darwiche. Compiling Bayesian networks using variable elimination. In *Proceedings of IJCAI 2007*, 2007.

[4] Gregory Cooper, Denver Dash, John Levander, Weng-Keen Wong, William Hogan, and Michael Wagner. Bayesian biosurveillance of disease outbreaks. In *Proceedings of UAI*, pages 94–103, 2004.

[5] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1990.

[6] Tal El-Hay, Nir Friedman, Daphne Koller, and Raz Kupferman. Continuous time Markov networks. In *Proceedings of UAI-06*. AUAI Press, 2006.

[7] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*, pages 35–45, March 1960.

[8] Jin H. Kim and Judea Pearl. A computational model for combined causal and diagnostic reasoning in inference systems. In *Proceedings IJCAI-83 (Karlsruhe, Germany)*, pages 190–193, 1983.

[9] Daphne Koller and Nir Friedman. *Bayesian Networks and Beyond*. Unpublished manuscript.

[10] David Larkin and Rina Dechter. Bayesian inference in the presence of determinism. In *Proceedings of Workshop on AI and Statistics, AISTAT 2003*.

[11] Kevin Murphy. The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*, 33, 2001.

[12] Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, EECS, University of California, Berkeley, Berkeley, CA, July 2002.

[13] Kevin Murphy and Yair Weiss. The factored frontier algorithm for approximate inference in DBNs. In *Proceedings of* 12^{th} *NIPS*, volume 12, 2000.

[14] Brenda M. Ng. Survey of Bayesian models for modelling of stochastic temporal processes. Technical Report UCRL-TR-225272, Lawrence Livermore National Laboratory, 2006.

[15] Mark Andrew Paskin. *Exploiting Locality in Probabilistic Inference*. PhD thesis, EECS, University of California, Berkeley, Berkeley, CA, December 2004.

[16] Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288, 1986.

[17] Mark A. Peot and Ross D. Shachter. Fusion and propagation with multiple observations in belief networks. *Artificial Intelligence*, 48(3):299–318, 1991.

[18] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, Feb 1989.

[19] Scott Sanner and David McAllester. Affine algebraic decision diagrams (AADDs) and their application to structured probabilistic inference. In *Proceedings of IJCAI 2005*, 2005.

[20] Tomáš Šingliar and Miloš Hauskrecht. Learning to detect adverse traffic events from noisily labeled data. In *Proceedings of PKDD 2007*, pages 236–247, 2007.

[21] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. In *Exploring Artificial Intelligence in the New Millennium*, January 2003.

Tightening LP Relaxations for MAP using Message Passing

David Sontag
CSAIL, MIT
Cambridge, MA

Talya Meltzer
Hebrew University
Jerusalem, Israel

Amir Globerson
CSAIL, MIT
Cambridge, MA

Tommi Jaakkola
CSAIL, MIT
Cambridge, MA

Yair Weiss
Hebrew University
Jerusalem, Israel

Abstract

Linear Programming (LP) relaxations have become powerful tools for finding the most probable (MAP) configuration in graphical models. These relaxations can be solved efficiently using message-passing algorithms such as belief propagation and, when the relaxation is tight, provably find the MAP configuration. The standard LP relaxation is not tight enough in many real-world problems, however, and this has lead to the use of higher order *cluster-based* LP relaxations. The computational cost increases exponentially with the size of the clusters and limits the number and type of clusters we can use. We propose to solve the cluster selection problem monotonically in the dual LP, iteratively selecting clusters with guaranteed improvement, and quickly re-solving with the added clusters by reusing the existing solution. Our dual message-passing algorithm finds the MAP configuration in protein side-chain placement, protein design, and stereo problems, in cases where the standard LP relaxation fails.

1 Introduction

The task of finding the maximum aposteriori assignment (or MAP) in a graphical model comes up in a wide range of applications. For an arbitrary graph, this problem is known to be NP hard [11] and various approximation algorithms have been proposed.

Linear Programming (LP) relaxations are commonly used to solve combinatorial optimization problems in computer science, and have a long history of being used to approximate the MAP problem in general graphical models (e.g., see [9]). LP relaxations have an advantage over other approximate inference schemes in that they come with an optimality guarantee – if the solution to the linear program is integral, then it is guaranteed to give the global optimum of the MAP problem.

An additional attractive quality of LP relaxations is that they can be solved efficiently using message-passing algorithms such as belief propagation and its generalizations [3, 13, 15]. In particular, by using message-passing algorithms, we can now use LP relaxations for large-scale problems where standard, off-the-shelf LP solvers could not be used [18].

Despite the success of LP relaxations, there are many real-world problems for which the basic LP relaxation is of limited utility in solving the MAP problem. For example, in a database of 97 protein design problems studied in [18], the standard LP relaxation allowed finding the MAP in only 2 cases.

One way to obtain tighter relaxations is to use *cluster-based* LP relaxations, where local consistency is enforced between cluster marginals. As the size of the clusters grow, this leads to tighter and tighter relaxations. Furthermore, message-passing algorithms can still be used to solve these cluster-based relaxations, with messages now being sent between clusters and not individual nodes. Unfortunately, the computational cost increases exponentially with the size of the clusters, and for many real-world problems this severely limits the number of large clusters that can be feasibly incorporated into the approximation. For example, in the protein design database studied in [18], each node has around 100 states, so even a cluster of only 3 variables would have 10^6 states. Clearly we cannot use too many such clusters in our approximation.

In this paper we propose a cluster-pursuit method where clusters are incrementally added to the relaxation, and where we only add clusters that are guaranteed to improve the approximation. Similar to the work of [16] who worked on region-pursuit for sum-product generalized belief propagation [19], we show

how to use the messages from a given cluster-based approximation to decide which cluster to add next. In addition, by working with a message-passing algorithm based on dual coordinate descent, we monotonically decrease an upper bound on the MAP value.

2 MAP and its LP Relaxation

We consider functions over n discrete variables $\boldsymbol{x} = \{x_1, \ldots, x_n\}$ defined as follows. Given a graph $G = (V, E)$ with n vertices, and potentials $\theta_{ij}(x_i, x_j)$ for all edges $ij \in E$, define the function

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{ij \in E} \theta_{ij}(x_i, x_j) + \sum_{i \in V} \theta_i(x_i) \ . \qquad (1)$$

Our goal is to find the MAP assignment, $\boldsymbol{x}_M$, that maximizes the function $f(\boldsymbol{x}; \boldsymbol{\theta})$.

The MAP problem can be formulated as a linear program as follows. Let $\boldsymbol{\mu}$ be a vector of marginal probabilities that includes $\{\mu_{ij}(x_i, x_j)\}_{ij \in E}$ over variables corresponding to edges and $\{\mu_i(x_i)\}_{i \in V}$ associated with the nodes. The set of $\boldsymbol{\mu}$ that arise from some joint distribution is known as the *marginal polytope* [14],

$$\mathcal{M}(G) = \left\{ \boldsymbol{\mu} \mid \exists p(\boldsymbol{x}) \text{ s.t. } \begin{array}{l} p(x_i, x_j) = \mu_{ij}(x_i, x_j) \\ p(x_i) = \mu_i(x_i) \end{array} \right\}.$$

The MAP problem can then be shown to be equivalent to the following LP,

$$\max_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{\theta}) = \max_{\boldsymbol{\mu} \in \mathcal{M}(G)} \boldsymbol{\mu} \cdot \boldsymbol{\theta} \ , \qquad (2)$$

where $\boldsymbol{\mu} \cdot \boldsymbol{\theta} = \sum_{ij \in E} \sum_{x_i, x_j} \theta_{ij}(x_i, x_j) \mu_{ij}(x_i, x_j) + \sum_i \sum_{x_i} \mu_i(x_i) \theta_i(x_i)$. There always exists a maximizing $\boldsymbol{\mu}$ that is integral – a vertex of the marginal polytope – and which corresponds to $\boldsymbol{x}_M$. Although the number of variables in this LP is only $O(|E|+|V|)$, the difficulty comes from an exponential number of linear inequalities typically required to describe the marginal polytope $\mathcal{M}(G)$.

The idea in LP relaxations is to relax the difficult global constraint that the marginals in $\boldsymbol{\mu}$ arise from some common joint distribution. Instead, we enforce this only over some subsets of variables that we refer to as clusters. More precisely, we introduce auxiliary distributions over clusters of variables and constrain the edge distributions $\mu_{ij}(x_i, x_j)$ associated with each cluster to arise as marginals from the cluster distribution.[1] Let $\mathcal{C}$ be a set of clusters such that each $c \in \mathcal{C}$ is a subset of $\{1, \ldots, n\}$, and let $\tau_c(x_c)$ be any distribution over the variables in c. We also use $\tau_c(x_i, x_j)$ to refer to the marginal of $\tau_c(x_c)$ for the edge (i, j), i.e. $\tau_c(x_i, x_j) = \sum_{x_{c \setminus i,j}} \tau_c(x_c)$. Define $\mathcal{M}_{\mathcal{C}}(G)$ as

$$\left\{ \begin{array}{l} \exists \boldsymbol{\tau} \geq 0 \\ \boldsymbol{\mu} \geq 0 \end{array} \middle| \begin{array}{ll} \sum_{x_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i) & \\ \tau_c(x_i, x_j) = \mu_{ij}(x_i, x_j) & \forall c, (i, j) \subseteq c \\ \sum_{x_c} \tau_c(x_c) = 1 & \end{array} \right\}$$

It is easy to see that $\mathcal{M}_{\mathcal{C}}(G)$ is an outer bound on $\mathcal{M}(G)$, namely $\mathcal{M}_{\mathcal{C}}(G) \supseteq \mathcal{M}(G)$. As we add more clusters to $\mathcal{C}$ the relaxation of the marginal polytope becomes tighter. Note that similar constraints should be imposed on the cluster marginals, i.e., they themselves should arise as marginals from some joint distribution. To exactly represent the marginal polytope, such a hierarchy of auxiliary clusters would require clusters of size equal to the treewidth of the graph. For the purposes of this paper, we will not generate such a hierarchy but instead use the clusters to constrain only the associated edge marginals.

2.1 Choosing Clusters in the LP Relaxation

Adding a cluster to the relaxation $\mathcal{M}_{\mathcal{C}}(G)$ requires computations that scale with the number of possible cluster states. The choice of clusters should therefore be guided by both how much we are able to constrain the marginal polytope, as well as the computational cost of handling larger clusters. We will consider a specific scenario where the clusters are selected from a pre-defined set of possible clusters $\mathcal{C}_0$ such as triplet clusters. However, we will ideally not want to use all of the clusters in $\mathcal{C}_0$, but instead add them gradually based on some ranking criterion.

The best ranking of clusters is problem dependent. In other words, we would like to choose the subset of clusters which will give us the best possible approximation to a particular MAP problem. We seek to *iteratively* improve the approximation, using our current beliefs to guide which clusters to add. The advantage of iteratively selecting the clusters is that we add them only up to the point that the relaxed LP has an integral solution.

Recently, Sontag and Jaakkola [12] suggested an approach for incrementally adding constraints to the marginal polytope using a *cutting-plane algorithm*. A similar approach may in principle be applied to adding clusters to the primal problem. One shortcoming of this approach is that it requires solving the primal LP after every cluster added, and even solving the primal LP once is infeasible for large problems involving hundreds of variables and large state spaces.

In the next section we present a method that incrementally adds clusters, but which works exclusively within the dual LP. The key idea is that the dual LP

[1] Each edge may participate in multiple clusters.

provides an upper bound on the MAP value, and we seek to choose clusters to most effectively minimize this bound. Note that an analogous bound minimization strategy is problematic in the primal where we would have to assess how much *less* the maximum is due to including additional constraints. In other words, obtaining a certificate for improvement is difficult in the primal. Moreover, unlike the dual, the primal algorithm might not give an upper bound on the MAP prior to convergence.

Finally, we can "warm start" our optimization scheme after each cluster addition in order to avoid re-solving the dual LP. We do this by reusing the dual variables calculated in the previous iterations which did not have the new clusters.

3 Dual LP Relaxation

The obstacles to working in the primal LP lead us to consider the dual of the LP relaxation. Different formulations of the primal LP have lead to different dual LPs, each with efficient message-passing algorithms for solving them [3, 6, 13, 15]. In this paper we focus on a particular dual formulation by Globerson and Jaakkola [3] which has the advantage that the message-passing algorithm corresponds to performing coordinate-descent in the dual LP. Our dual algorithm will address many of the problems that were inherent in the primal approaches, giving us:

1. Monotonically decreasing upper bound on MAP.
2. Choosing clusters which give a guaranteed bound improvement.
3. Simple "warm start" of tighter relaxation.
4. An efficient algorithm that scales to very large problems.

3.1 The Generalized MPLP Algorithm

The generalized Max-Product LP (MPLP) message-passing algorithm, introduced in [3], decreases the dual objective of the cluster-based LP relaxation at every iteration. This monotone property makes it ideal for adding clusters since we can initialize the new messages such that the dual value is monotonically decreased.

Another key advantage of working in the dual is that the dual objective gives us a certificate of optimality. Namely, if we find an assignment $\boldsymbol{x}$ such that $f(\boldsymbol{x};\boldsymbol{\theta})$ is equal to the dual objective, we are guaranteed that $\boldsymbol{x}$ is the MAP assignment (since the dual objective upper bounds the MAP value). Indeed, using this property we show in our experiments that MAP assignments can be found for nearly all of the problems we consider.

We next describe the generalized MPLP algorithm for the special case of clusters comprised of three nodes. Although the algorithm applies to general clusters, we focus on triplets for simplicity, and because these are the clusters used in the current paper.

MPLP passes the following types of messages:

- **Edge to Node**: For every edge $e \in E$ (e denotes two indices in V) and every node $i \in e$, we have a message $\lambda_{e\to i}(x_i)$.
- **Edge to Edge**: For every edge $e \in E$, we have a message $\lambda_{e\to e}(x_e)$ (where x_e is shorthand for x_i, x_j, and i and j are the nodes in the edge).
- **Triplet to Edge**: For every triplet cluster $c \in \mathcal{C}$, and every edge $e \in c$, we have a message $\lambda_{c\to e}(x_e)$.

The updates for these messages are given in Figure 1. To guarantee that the dual objective decreases, all messages from a given edge must be sent simultaneously, as well as all messages from a triplet to its three edges.

The dual objective that is decreased in every iteration is given by

$$\begin{aligned} g(\boldsymbol{\lambda}) &= \sum_{i\in V} \max_{x_i} \left[\theta_i(x_i) + \sum_{k\in N(i)} \lambda_{ki\to i}(x_i)\right] \\ &+ \sum_{e\in E} \max_{x_e} \left[\lambda_{e\to e}(x_e) + \sum_{c:e\in c} \lambda_{c\to e}(x_e)\right] \end{aligned}$$

It should be noted, however, that not all $\boldsymbol{\lambda}$ are dual feasible. Rather, $\boldsymbol{\lambda}$ needs to result from a reparameterization of the underlying potentials (see [3]). However, it turns out that after updating all the MPLP messages once, all subsequent $\boldsymbol{\lambda}$ will be dual feasible, regardless of how $\boldsymbol{\lambda}$ is initialized.[2]

By LP duality, there exists a value of $\boldsymbol{\lambda}$ such that $g(\boldsymbol{\lambda})$ is equal to the optimum of the corresponding primal LP. Although the MPLP updates decrease the objective at every iteration, they may converge to a $\boldsymbol{\lambda}$ that is not dual optimal, as discussed in [3]. However, as we will show in the experiments, our procedure often finds the exact MAP solution, and therefore also achieves the primal optimum in these cases.

3.2 Choosing Clusters in the Dual LP Relaxation

In this section we provide a very simple procedure that allows adding clusters to MPLP, while satisfying the

[2]In our experiments, we initialize all messages to zero.

- **Edge to Node**: For every edge $ij \in E$ and node i (or j) in the edge:

$$\lambda_{ij\to i}(x_i) \leftarrow -\frac{2}{3}\Big(\lambda_i^{-j}(x_i)+\theta_i(x_i)\Big)+\frac{1}{3}\max_{x_j}\Big[\sum_{c:ij\in c}\lambda_{c\to ij}(x_i,x_j)+\lambda_j^{-i}(x_j)+\theta_{ij}(x_i,x_j)+\theta_j(x_j)\Big]$$

  where $\lambda_i^{-j}(x_i)$ is the sum of edge-to-node messages into i that are not from edge ij, namely: $\lambda_i^{-j}(x_i) = \sum_{k\in N(i)\setminus j}\lambda_{ik\to i}(x_i)$.

- **Edge to Edge**: For every edge $ij \in E$:

$$\lambda_{ij\to ij}(x_i,x_j) \leftarrow -\frac{2}{3}\sum_{c:ij\in c}\lambda_{c\to ij}(x_i,x_j)+\frac{1}{3}\Big[\lambda_j^{-i}(x_j)+\lambda_i^{-j}(x_i)+\theta_{ij}(x_i,x_j)+\theta_i(x_i)+\theta_j(x_j)\Big]$$

- **Triplet to Edge**: For every triplet $c \in \mathcal{C}$ and every edge $e \in c$:

$$\lambda_{c\to e}(x_e) \;\leftarrow\; -\frac{2}{3}\Big(\lambda_{e\to e}(x_e)+\sum_{\substack{c'\neq c\\ e\in c'}}\lambda_{c'\to e}(x_e)\Big)+\frac{1}{3}\max_{x_{c\setminus e}}\Big[\sum_{e'\in c\setminus e}\Big(\lambda_{e'\to e'}(x_{e'})+\sum_{\substack{c'\neq c\\ e'\in c'}}\lambda_{c'\to e'}(x_{e'})\Big)\Big]$$

Figure 1: The generalized MPLP updates for an LP relaxation with three node clusters.

algorithmic properties in the beginning of Section 3.

Assume we have a set of triplet clusters $\mathcal{C}$ and now wish to add a new triplet. Denote the messages before adding the new triplet by $\boldsymbol{\lambda}_t$. Two questions naturally arise. The first is: assuming we decide to add a given triplet, how do we set $\boldsymbol{\lambda}_{t+1}$ such that the dual objective retains its previous value $g(\boldsymbol{\lambda}_t)$. The second question is how to choose the new triplet to add.

The initialization problem is straightforward. Simply set $\boldsymbol{\lambda}_{t+1}$ to equal $\boldsymbol{\lambda}_t$ for all messages from triplets and edges in the previous run, and set $\boldsymbol{\lambda}_{t+1}$ for the messages from the new triplet to its edges to zero.[3] This clearly results in $g(\boldsymbol{\lambda}_{t+1}) = g(\boldsymbol{\lambda}_t)$.

In order to choose a *good* triplet, one strategy would be to add different triplets and run MPLP until convergence to find the one that decreases the objective the most. However, this may be computationally costly and, as we show in the experiments, is not necessary. Instead, the criterion we use is to consider the decrease in value that results from just sending messages from the triplet c to its edges (while keeping all other messages fixed).

The decrease in $g(\boldsymbol{\lambda})$ resulting from such an update has a simple form, as we show next. Assume we are considering adding a triplet c. For every edge $e \in c$, define $b_e(x_e)$ to be

$$b_e(x_e) = \lambda_{e\to e}(x_e)+\sum_{c':e\in c'}\lambda_{c'\to e}(x_e)\ , \qquad (3)$$

where the summation over clusters c' does not include c (those messages are initially zero). The decrease in $g(\boldsymbol{\lambda})$ corresponding to updating only messages from c to the edges $e \in c$ can be shown to be

$$d(c) = \sum_{e\in c}\max_{x_e} b_e(x_e) - \max_{x_c}\left[\sum_{e\in c} b_e(x_e)\right]\ . \qquad (4)$$

The above corresponds to the difference between independently maximizing each edge and jointly maximizing over the three edges. Thus $d(c)$ is a lower bound on the improvement in the dual objective if we were to add triplet c. Our algorithm will therefore add the triplet c that maximizes $d(c)$.

3.3 The Dual Algorithm

We now present the complete algorithm for adding clusters and optimizing over them. Let $\mathcal{C}_0$ be the predefined set of triplet clusters that we will consider adding to our relaxation, and let $\mathcal{C}_L$ be the initial relaxation consisting of only edge clusters (pairwise local consistency).

1. Run MPLP until convergence using the $\mathcal{C}_L$ clusters.

2. Find an integral solution $\boldsymbol{x}$ by locally maximizing the single node beliefs $b_i(x_i)$, where $b_i(x_i) = \theta_i(x_i) + \sum_{k\in N(i)}\lambda_{ki\to i}(x_i)$. Ties are broken arbitrarily.

3. If the dual objective $g(\boldsymbol{\lambda}_t)$ is sufficiently close to the primal objective $f(\boldsymbol{x};\theta)$, terminate (since $\boldsymbol{x}$ is approximately the MAP).

[3] It is straightforward to show that $\boldsymbol{\lambda}_{t+1}$ is dual feasible.

4. Add the cluster $c \in \mathcal{C}_0$ with the largest guaranteed bound improvement, $d(c)$, to the relaxation.

5. Construct "warm start" messages $\boldsymbol{\lambda}_{t+1}$ from $\boldsymbol{\lambda}_t$.

6. Run MPLP for N iterations, and return to **2**.

Note that we obtain (at least) the promised bound improvement $d(c)$ within the first iteration of step **6**. By allowing MPLP to run for N iterations, the effect of adding the cluster will be propagated throughout the model, obtaining an additional decrease in the bound. Since the MPLP updates correspond to coordinate-descent in the dual LP, every step of the algorithm decreases the upper bound on the MAP. The monotonicity property holds even if MPLP does not converge in step **6**, giving us the flexibility to choose the number of iterations N. In Section 5 we show results corresponding to two different choices of N.

In the case where we run MPLP to convergence before choosing the next cluster, we can show that the greedy bound minimization corresponds to a cutting-plane algorithm, as stated below.

Theorem 1. *Given a dual optimal solution, if we find a cluster for which we can guarantee a bound decrease, all primal optimal solutions were inconsistent with respect to this cluster.*

Proof. By duality both the dual optimum and the primal optimum will decrease. Suppose for contradiction that in the previous iteration there was a primal feasible point that was cluster consistent and achieved the LP optimum. Since we are maximizing the LP, after adding the cluster consistency constraint, this point is still feasible and the optimal value of the primal LP will not change, giving our contradiction. □

This theorem does not tell us *how much* the given cluster consistency constraint was violated, and the distinction remains that a typical cutting-plane algorithm would attempt to find the constraint which is most violated.

4 Related Work

Since MPLP is closely related to the max-product generalized belief propagation (GBP) algorithm, our work can be thought of as a region-pursuit method for GBP. This is closely related to the work of Welling [16] who suggested a region-pursuit method for sum-product GBP. Similar to our work, he suggested greedily adding from a candidate set of possible clusters. At each iteration, the cluster that results in the largest change in the GBP free energy is added. He showed excellent results for 2D grids, but on fully connected graphs the performance actually started deteriorating with additional clusters. In [17], a heuristic related to maxent normality [19] was used as a stopping criterion for region-pursuit to avoid this behavior. In our work, in contrast, since we are working with the dual function of the LP, we can guarantee monotonic improvement throughout the running of the algorithm.

Our work is also similar to Welling's in that we focus on criteria for determining the utility of adding a cluster, not on *finding* these clusters efficiently. We found in our experiments that a simple enumeration over small clusters proved extremely effective. For problems where triplet clusters alone would not suffice to find the MAP, we could triangulate the graph and consider larger clusters. This approach is reminiscent of the bounded join-graphs described in [1].

There is a large body of recent work describing the relationship between message-passing algorithms such as belief propagation, and LP relaxations [7, 15, 18]. Although we have focused here on using one particular message-passing algorithm, MPLP, we emphasize that similar region-pursuit algorithms can be derived for other message-passing algorithms as well. In particular, for all the convex max-product BP algorithms described in [15], it is easy to design region-pursuit methods. The main advantage of using MPLP is its guaranteed decrease of the dual value at each iteration, a guarantee that does not exist for general convex BP algorithms.

Region-pursuit algorithms are also conceptually related to the question of message scheduling in BP, as in the work of Elidan et al. [2]. One way to think of region-pursuit is to consider a graph where all the clusters are present all the time, but send and receive non-informative messages. The question of which cluster to add to an approximation, is thus analogous to the question of which message to update next.

5 Experiments

Due to the scalable nature of our message-passing algorithm, we can apply it to cases where standard LP solvers cannot be applied to the primal LP (see also [18]). Here we report applications to problems in computational biology and machine vision.[4]

We use the algorithm from Section 3.3 for all of our experiments. We first run MPLP with edge clusters until convergence or for at most 1000 iterations, whichever comes first. All of our experiments, except those intended to show the difference between schedules, use $N = 20$ for the number of MPLP iterations run after adding a cluster. While running MPLP we use the messages to decode an integral solution, and compare

[4] Graphical models for these are given in [18].

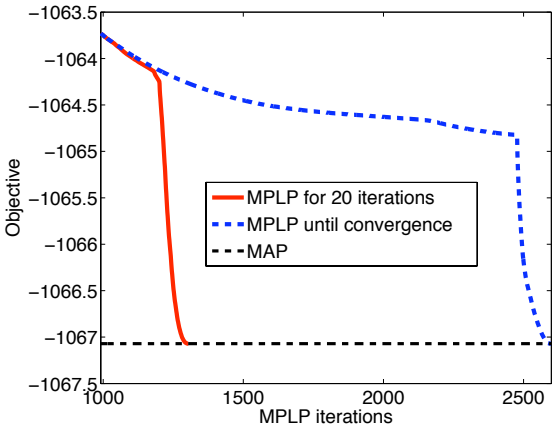


Figure 2: Comparison of different schedules for adding clusters to relaxation on a side-chain prediction problem.

the dual objective to the value of the integral solution. If these are equal, we have found the MAP solution.[5] Otherwise, we keep adding triplets.

Our results will show that we often find the MAP solution to these hard problems by using only a small number of triplet clusters. This indicates both that triplets are sufficient for characterizing $\mathcal{M}(G)$ near the MAP solution of these problems, and that our algorithm can efficiently find the informative triplets.

5.1 Side-Chain Prediction

The side-chain prediction problem involves finding the three-dimensional configuration of rotamers given the backbone structure of a protein [18]. This problem can be posed as finding the MAP configuration of a pairwise model, and in [18] the TRBP algorithm [13] was used to find the MAP solution for most of the models studied. However, for 30 of the models, TRBP could not find the MAP solution.

In earlier work [12] we used a cutting-plane algorithm to solve these side-chain problems and found the MAP solution for all 30 models. Here, we applied our dual algorithm to the same 30 models and found that it also results in the MAP solution for all of them (up to a 10^{-4} integrality gap). This required adding between 1 and 27 triplets per model. The running time was between 1 minute and 1 hour to solve each problem, with over half solved in under 9 minutes. On average we added only 7 triplets (median was 4.5), another indication of the relative ease with which these techniques can solve the side-chain prediction problem.

[5]In practice, we terminate when the dual objective is within 10^{-4} of the decoded assignment, so these are approximate MAP solutions. Note that the objective values are significantly larger than this threshold.

We also used these models to study different update schedules. One schedule (which gave the results in the previous paragraph) was to first run a pairwise model for 1000 iterations, and then alternate between adding triplets and running MPLP for 20 more iterations. In the second schedule, we run MPLP to convergence after adding each triplet. Figure 2 shows the two schedules for the side-chain protein '1gsk', one of the side-chain proteins which took us the longest to solve (30 minutes). Running MPLP to convergence results in a much larger number of overall MPLP iterations compared to using only 20 iterations. This highlights one of the advantages of our method: adding a new cluster does not require solving the earlier problem to convergence.

5.2 Protein Design

The protein design problem is the inverse of the protein folding problem. Given a particular 3D shape, we wish to find a sequence of amino-acids that will be as stable as possible in that 3D shape. Typically this is done by finding a set of amino-acids and rotamer configurations that minimizes an approximate energy.

While the problem is quite different from side-chain prediction, it can be solved using the same graph structure, as shown in [18]. The only difference is that now the nodes do not just denote rotamers, but also the identity of the amino-acid at that location. Thus, the state-space here is significantly larger than in the side-chain prediction problem (up to 180 states per variable for most variables).

In contrast to the side-chain prediction problems, which are often easily solved by general purpose integer linear programming packages such as CPLEX's branch-and-cut algorithm [5], the sheer size of the protein design problems immediately limits the techniques by which we can attempt to solve them. Algorithms such as our earlier cutting-plane algorithm [12] or CPLEX's branch-and-cut algorithm require solving the primal LP relaxation at least once, but solving the primal LP on all but the smallest of the design problems is intractable [18]. Branch and bound schemes have been recently used in conjunction with a message passing algorithm [4] and applied to similar protein design problems, although not the ones we solve here.

We applied our method to the 97 protein design problems described in [18], adding 5 triplets at a time to the relaxation. The key striking result of these experiments is that our method found the exact MAP configuration for all but one of the proteins[6] (up to a precision of 10^{-4} in the integrality gap). This is es-

[6]We could not solve '1fpo', the largest protein.

pecially impressive since, as reported in [18], only 2 of these problems were solvable using TRBP, and the primal problem was too big for commercial LP solvers such as CPLEX. For the problem where we did not find the MAP, we did not reach a point where all the triplets in the graph were included, since we ran out of memory beforehand.

Among the problems that were solved exactly, the mean running time was 9.7 hours with a maximum of 11 days and a minimum of a few minutes. We note again that most of these problems could not be solved using LP solvers, and when LP solvers could be used, they were typically at least 10 times slower than message-passing algorithms similar to ours (see [18] for detailed timing comparisons).

Note that the main computational burden in the algorithm is processing triplet messages. Since each variable has roughly 100 states, passing a triplet message requires 10^6 operations. Thus the number of triplets added is the key algorithmic complexity issue. For the models that were solved exactly, the median number of triplets added was 145 (min: 5, max: 735). As mentioned earlier, for the unsolved model this number grew until the machine's memory was exhausted. We believe however, that by optimizing our code for speed and memory we will be able to accommodate a larger number of triplets, and possibly solve the remaining model as well. Our current code is written mostly in Matlab, so significant optimization may be possible.

5.3 Stereo Vision

Given a stereo pair of images, the stereo problem is to find the disparity of each pixel in a reference image. This disparity can be straightforwardly translated into depth from the camera. The best algorithms currently known for the stereo problem are those that minimize a global energy function [10], which is equivalent to finding a MAP configuration in a pairwise model.

For our experiments we use the pairwise model described in [18], and apply our procedure to the "Tsukuba" sequence from the standard Middlebury stereo benchmark set [10], reduced in size to contain 116x154 pixels.

Since there are no connected triplets in the grid graph, we use our method with square clusters. We calculate the bound decrease using square clusters, but rather than add them directly, we triangulate the cycle and add two triplet clusters. This results in an equivalent relaxation, but has the consequence that we may have to wait until MPLP convergence to achieve the guaranteed bound improvement.

In the first experiment, we varied the parameters of the energy function to create several different instances. We tried to find the MAP using TRBP, resolving ties using the methods proposed in [8]. In 4 out of 10 cases those methods failed. Using our algorithm, we managed to find the MAP for all 4 cases.[7]

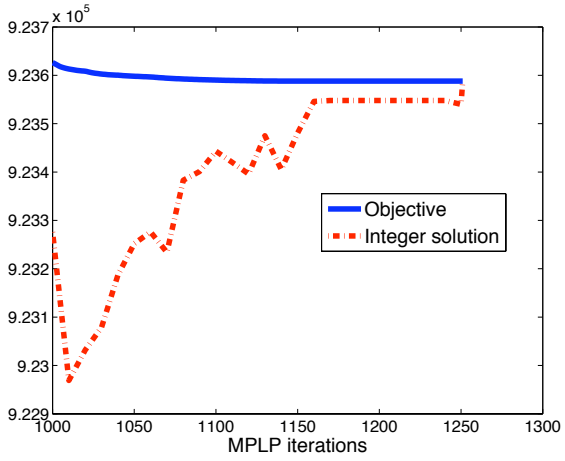


Figure 3: Dual objective and value of decoded integer solution for one of the reduced "Tsukuba" stereo models, as a function of MPLP iterations. It can be seen that both curves converge to the same value, indicating that the MAP solution was found.

Figure 3 shows the dual objective and the decoded integer solution after each MPLP iteration, for one set of parameters.

In the results above, we added 20 squares at a time to the relaxation. We next contrasted it with two strategies: one where we pick 20 random squares (not using our bound improvement criterion) and one where we pick the single best square according to the bound criterion. Figure 4 shows the resulting bound per iteration for one of the models. It can be seen that the random method is much slower than the bound criterion based one, and that adding 20 squares at a time is better than just one. We ended up adding 1060 squares when adding 20 at a time, and 83 squares when adding just one. Overall, adding 20 squares at a time turned out to be faster.

We also tried running MPLP with all of the square clusters. Although fewer MPLP iterations were needed, the cost of using all squares resulted in an overall running time of about four times longer.

[7] For one of these models, a few single node beliefs at convergence were tied, and we used the junction tree algorithm to decode the tied nodes (see [8]).

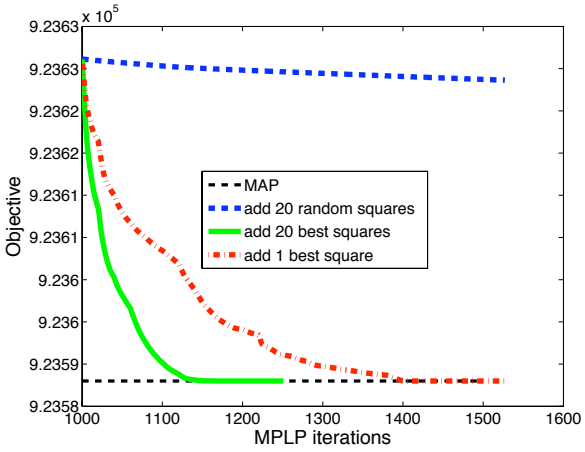


Figure 4: Comparison of different schedules for adding squares in one of the stereo problems.

6 Conclusion

In order to apply LP relaxations to real-world problems, one needs to find an efficient way of adding clusters to the basic relaxation such that the problem remains tractable but yields a better approximation of the MAP value.

In this paper we presented a greedy bound-minimization algorithm on the dual LP to solve this problem, and showed that it has all the necessary ingredients: an efficient message-passing algorithm, "warm start" of the next iteration using current beliefs, and a monotonically decreasing bound on the MAP.

We showed that the algorithm works well in practice, finding the MAP configurations for many real-world problems that were previously thought to be too difficult for known methods to solve. While in this paper we focused primarily on adding triplet clusters, our approach is general and can be used to add larger clusters as well, as long as as the messages in the dual algorithm can be efficiently computed.

Finally, while here we focused on the MAP problem, similar ideas may be applied to approximating the marginals in graphical models.

Acknowledgements

This work was supported in part by the DARPA Transfer Learning program and by the Israeli Science Foundation. D.S. was also supported by a NSF Graduate Research Fellowship.

References

[1] R. Dechter, K. Kask, and R. Mateescu. Iterative join-graph propagation. In *UAI*, 2002.

[2] G. Elidan, I. Mcgraw, and D. Koller. Residual belief propagation: informed scheduling for asynchronous message passing. In *UAI*, 2006.

[3] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Advances in Neural Information Processing Systems 21*. MIT Press, 2008.

[4] E.J. Hong and T. Lozano-Pérez. Protein side-chain placement through MAP estimation and problem-size reduction. In *WABI*, 2006.

[5] C. L. Kingsford, B. Chazelle, and M. Singh. Solving and analyzing side-chain positioning problems using linear and integer programming. *Bioinformatics*, 21(7):1028–1039, 2005.

[6] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1568–1583, 2006.

[7] V. Kolmogorov and M. Wainwright. On the optimality of tree-reweighted max-product message-passing. In *UAI*, 2005.

[8] T. Meltzer, C. Yanover, and Y. Weiss. Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *ICCV*, 2005.

[9] E.G. Santos. On the generation of alternative explanations with implications for belief revision. In *UAI*, 1991.

[10] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002.

[11] Y. Shimony. Finding the MAPs for belief networks is NP-hard. *Aritifical Intelligence*, 68(2):399–410, 1994.

[12] D. Sontag and T. Jaakkola. New outer bounds on the marginal polytope. In *Advances in Neural Information Processing Systems 21*. MIT Press, 2008.

[13] M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Trans. on Information Theory*, 51(11):3697–3717, 2005.

[14] M. Wainwright and M. I. Jordan. Graphical models, exponential families and variational inference. Technical report, UC Berkeley, Dept. of Statistics, 2003.

[15] Y. Weiss, C. Yanover, and T. Meltzer. MAP estimation, linear programming and belief propagation with convex free energies. In *UAI*, 2007.

[16] M. Welling. On the choice of regions for generalized belief propagation. In *UAI*, 2004.

[17] M. Welling, T. Minka, and Y. W. Teh. Structured region graphs: Morphing EP into GBP. In *UAI*, 2005.

[18] C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation – an empirical study. *JMLR*, 7:1887–1907, 2006.

[19] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Trans. on Information Theory*, 51(7):2282– 2312, 2005.

Learning the Bayesian Network Structure: Dirichlet Prior versus Data

Harald Steck
Computer-Aided Diagnosis (IKM CKS)
Siemens Medical Solutions, 51 Valley Stream Parkway E51
Malvern, PA 19355, USA
harald.steck@siemens.com

Abstract

In the Bayesian approach to structure learning of graphical models, the equivalent sample size (ESS) in the Dirichlet prior over the model parameters was recently shown to have an important effect on the maximum-a-posteriori estimate of the Bayesian network structure. In our first contribution, we theoretically analyze the case of *large* ESS-values, which complements previous work: among other results, we find that the presence of an edge in a Bayesian network is favoured over its absence even if both the Dirichlet prior and the data imply independence, as long as the conditional empirical distribution is notably different from uniform. In our second contribution, we focus on realistic ESS-values, and provide an analytical approximation to the 'optimal' ESS-value in a predictive sense (its accuracy is also validated experimentally): this approximation provides an understanding as to which properties of the data have the main effect determining the 'optimal' ESS-value.

1 INTRODUCTION

The posterior probability and the marginal likelihood of the graph are one of the most popular scoring functions for learning Bayesian network structures, e.g. [Buntine, 1991, Heckerman *et al.*, 1995]. Both of them require the value of a free parameter to be specified by the researcher: the so-called *equivalent sample size* (ESS), which originates from the Dirichlet prior over the model parameters, cf. [Buntine, 1991, Heckerman *et al.*, 1995]. In the elaborate experiments on 20 UCI data sets in [Silander *et al.*, 2007], it was shown that the chosen ESS-value has a decisive effect on the resulting maximum-a-posteriori (MAP) graph estimate: in the experiment shown in Figure 1 in [Silander *et al.*, 2007], the number of edges increases monotonically from zero to the maximum as the ESS-value grows. These two extremes (empty and complete graph) are indeed reached for several data sets, and they are almost reached for the remaining data sets in [Silander *et al.*, 2007].

The effect of the ESS-value on the MAP graph estimate was earlier noticed in [Steck & Jaakkola, 2002], and one of its main contributions was the theoretical analysis of the case of *small* ESS-values. That paper also provided a result for the case ESS$\rightarrow\infty$: the Bayes factor converges to zero *in the limit*, which means that it favours neither the presence nor the absence of an edge. Given this inconclusive behavior *in the limit*, our first contribution in this paper is concerned with the case of *large but finite* ESS-values: in our theoretical analysis in Section 3, we derive the properties of the given data under which the presence of an edge in the graph is favoured. This contribution concerning the case of large but finite ESS-values, combined with the results for small ESS-values [Steck & Jaakkola, 2002], implies immediately that, under these conditions, the number of edges increases when the ESS-value grows, although not necessarily monotonically. Our second contribution is concerned with the case of *realistic / intermediate* ESS-values (Section 4): our goal is to *understand* as to which properties of a given data set determine the 'optimal' ESS-value (in the predictive sense). We achieve this by deriving an explicit analytical approximation (its validity is also assessed experimentally).

2 BRIEF REVIEW OF BAYESIAN NETWORK LEARNING

A Bayesian network model comprises a directed acyclic graph G, and the model parameters θ are conditional probabilities, e.g., see [Cowell *et al.*, 1999] for an overview. It describes a joint probability distribution $p(X|G)$ over the vector of random variables $X = (X_1, ..., X_n)$, where n is the number of variables. We consider discrete random variables X_i with a multinomial distribution in this paper. According to the graph G, the joint distribution factors recursively, $p(X|G) = \prod_{i=1}^{n} \theta_{X_i|\Pi_i}$; Π_i denotes the set of parents

of variable/node X_i in the graph G; the (joint) states of Π_i will be denoted by π_i, and the states of X_i by x_i.

Various approaches to learning the graph G from given data D have been developed in the literature, e.g., see [Cowell *et al.*, 1999] for an overview. The Bayesian approach appears to be most popular. As a scoring function, it employs the posterior probability of the graph G given the data, $p(G|D) \propto p(D|G)p(G)$, or the marginal likelihood of the graph, $p(D|G)$. Both are equivalent if the prior distribution over the graphs, $p(G)$, is chosen to be uniform. The marginal likelihood of the graph can be expressed in closed form when using the Dirichlet prior over the model parameters [Buntine, 1991, Heckerman *et al.*, 1995]. More importantly, the Dirichlet prior is the only distribution that ensures likelihood equivalence [Heckerman *et al.*, 1995], a desirable property in structure learning. In particular, likelihood equivalence holds only if the hyper-parameters α_{x_i,π_i} of the Dirichlet prior can be expressed as $\alpha_{x_i,\pi_i} = \alpha \cdot q_{x_i,\pi_i}$ for all x_i, π_i and for all $i = 1,...,n$; q is a prior distribution over X, and α is a positive constant independent of i, the so-called scale-parameter or *equivalent sample size* (ESS). When q is chosen to be a uniform distribution, one arrives at the BDeu score [Buntine, 1991]: $q_{x_i,\pi_i} = 1/(|X_i| \cdot |\Pi_i|)$, where $|\cdot|$ denotes the number of (joint) states of the variable(s). Hence,

$$\alpha_{x_i,\pi_i} = \frac{\alpha}{|X_i| \cdot |\Pi_i|}, \tag{1}$$

so that the ESS α is the only remaining free parameter in the marginal likelihood of the graph G: $p(D|\alpha, G)$ equals [Buntine, 1991, Heckerman *et al.*, 1995]

$$\prod_{i=1}^{n} \prod_{\pi_i} \frac{\Gamma(\alpha_{\pi_i})}{\Gamma(N_{\pi_i} + \alpha_{\pi_i})} \prod_{x_i} \frac{\Gamma(N_{x_i,\pi_i} + \alpha_{x_i,\pi_i})}{\Gamma(\alpha_{x_i,\pi_i})}, \tag{2}$$

where Γ is the Gamma function; the first product extends over all the variables, while the second one is over all the joint states π_i of the parents Π_i of variable X_i, and the third product is over all the states x_i of X_i. The cell counts N_{x_i,π_i} in the contingency tables serve as sufficient statistics; the sample size is $N = \sum_{x_i,\pi_i} N_{x_i,\pi_i}$. In this Bayesian approach with the Dirichlet prior, the regularized parameter estimates are [Buntine, 1991]

$$\bar{\theta}_{x_i|\pi_i} \equiv E_{p(\theta_{x_i|\pi_i}|D,G)}[\theta_{x_i|\pi_i}] = \frac{N_{x_i,\pi_i} + \alpha_{x_i,\pi_i}}{N_{\pi_i} + \alpha_{\pi_i}}, \tag{3}$$

which is the expectation with respect to the parameter's posterior distribution.

For fixed data D and ESS α, finding the maximum-a-posteriori (MAP) estimate of the graph G, or the maximum with respect to the marginal likelihood in Eq. 2, is an NP-complete problem [Chickering, 1996]. One thus has to resort to heuristic search strategies, like local search (e.g., [Heckerman *et al.*, 1995]), as to find a close-to-optimal graph. If the number of variables is not prohibitively large, the exact solution, i.e., the globally optimal graph G, can be found in reasonable computation-time due to recent advances in structure learning [Silander & Myllymäki, 2006, Koivisto & Sood, 2004].

Besides this popular Bayesian score, various other scoring functions have been devised for structure learning, including the Bayesian information criteria (BIC) [Schwarz, 1978, Haughton, 1988], the Akaike Information Criteria (AIC) [Akaike, 1973], and the Minimum Description Length (MDL) [Rissanen, 1978]. We will use the AIC in our second contribution in Section 4.2.

3 LARGE ESS-VALUE: ITS EFFECT ON THE LEARNED GRAPH

In this section we present our first contribution, namely as to how a *large but finite* value of the ESS affects the learned optimal Bayesian network structure. This complements the theoretical results for *small* ESS-values derived in [Steck & Jaakkola, 2002] and the experimental results for *'intermediate'* ESS-values obtained in [Silander *et al.*, 2007].

3.1 UNIFORMITY-MEASURE

In this section, we define a new 'uniformity' measure and discuss its properties. This measure (and its properties) determines the result of structure learning for large ESS values, as we will see in the next section.

Definition: *We define the uniformity-measure U of a conditional multivariate distribution $p(A,B|\Pi)$ over two random variables A and B given a set of variables Π, as follows:*

$$\begin{aligned} &U(p(A,B|\Pi)) \\ &= \sum_{a,b,\pi} p(a,b,\pi)\Big(|A,B,\Pi| \cdot p(a,b,\pi) - |A,\Pi| \cdot p(a,\pi) \\ &\qquad -|B,\Pi| \cdot p(b,\pi) + |\Pi| \cdot p(\pi)\Big), \end{aligned} \tag{4}$$

where $|A,B,\Pi|$ denotes the number of joint states.

Obviously, this definition is equivalent to:

$$\begin{aligned} U(p(A,B|\Pi)) &= |\Pi| \cdot \sum_{\pi} p(\pi)^2 \sum_{a,b} p(a,b|\pi) \\ &\quad \cdot \Big(|A,B| \cdot p(a,b|\pi) - |A| \cdot p(a|\pi) - |B| \cdot p(b|\pi) + 1\Big) \\ &= |A,B,\Pi| \cdot \sum_{a,b,\pi} p(a,b,\pi)^2 - |A,\Pi| \cdot \sum_{a,\pi} p(a,\pi)^2 \\ &\quad -|B,\Pi| \cdot \sum_{b,\pi} p(b,\pi)^2 + |\Pi| \cdot \sum_{\pi} p(\pi)^2, \end{aligned} \tag{5}$$

where U is rewritten in terms of conditional probabilities in the first line, and in terms of squared probabilities in the second line.

The interesting property of U (in each representation) is that it is a weighted sum of four terms, where the weights are the number of (joint) states of the variables.

Proposition 1: *The measure U has the following three basic properties:*

- *symmetry:* $U(p(A,B|\Pi)) = U(p(B,A|\Pi))$
- *non-negativity:* $U(p(A,B|\Pi)) \geq 0$
- *minimality:* $U(p(A,B|\Pi)) = 0$ *if and only if*
  - *(conditional) independence:* $p(A,B|\Pi) = p(A|\Pi)p(B|\Pi)$
  - **and**, *for each state π with $p(\pi) > 0$, at least one of the marginal distributions is uniform:* $p(A|\pi) = 1/|A|$ *or* $p(B|\pi) = 1/|B|$.

Concerning the minimality, note that $U > 0$ if a state π with $p(\pi) > 0$ exists such that neither one of $p(A|\pi)$ or $p(B|\pi)$ is uniform: this includes distributions where A and B are conditionally independent, but both distributions are skewed. In other words, U is not necessarily equal to zero for independent variables.

Proof: The symmetry is obvious. As to proof the other two properties, we focus on the representation in terms of conditional probabilities in Eq. 5; obviously, if these properties hold for each state π, then they hold also for U, the sum. As the remainder of the proof is understood to be conditional on a state π with $p(\pi) > 0$, we now omit π from the notation. A necessary condition for minimization of U under the normalization-constraint $\sum_{a,b} p(a,b) = 1$ (accounted for by introducing the Lagrange multiplier λ), is that all the first partial derivatives vanish, i.e., for *all* states a,b:

$$2|A,B|p(a,b) - 2|A|p(a) - 2|B|p(b) + \lambda = 0 \qquad (6)$$

With the normalization constraint it follows immediately that $\lambda = 2$. It follows for the particular choice of considering the difference between each pair of these equations pertaining to the same state b, but different states a and a':

$$p(a,b) - p(a',b) = [p(a) - p(a')]/|B|$$

Hence, if $p(a) = p(a')$ for all a,a', then $p(A)$ is uniform and hence $p(A,B) = p(A)p(B)$ with arbitrary $p(B)$. Otherwise, i.e., if $p(a) \neq p(a')$ for some a,a', then $p(A,B) = p(A)p(B)$, where $p(B)$ has to be uniform, and $p(A)$ can be arbitrary. Conversely, it can be verified easily that such a distribution indeed fulfills the original condition in Eq. 6. Moreover, $U = 0$ for such a distribution. Finally, it can be shown that the second derivative is positive definite, which completes the proof. □

Aside: Besides the interesting fact that U is a weighted sum where the number of states function as the weights, we also like to mention that each of the four terms themselves is related to well-known quantities: the squared probabilities in Eq. 5 suggest a relationship to the well-known Gini index, $H^{\mathrm{G}}(p(X)) = 1 - \sum_x p_x^2$, which is used as an impurity measure when learning decision trees. It can also be viewed as a special case of the Tsallis entropy $H^{\mathrm{T}}_\beta(p(X)) = (1 - \sum_x p_x^\beta)/(\beta - 1)$ with parameter $\beta = 2$ [Tsallis, 2000]. The Tsallis entropy is used in statistical physics, and can be understood as the leading-order Taylor expansion of the Renyi entropy, $H^{\mathrm{R}}_\beta(p(X)) = (\log \sum_x p_x^\beta)/(1-\beta)$, which is a generalization of the Shannon entropy. In the limit $\beta \to 1$, the Tsallis entropy and the Renyi entropy both coincide with the Shannon entropy.

3.2 LEADING-ORDER APPROXIMATION OF BAYES FACTOR

In this section, we theoretically analyze the behavior of the Bayesian approach to structure learning, as reviewed in Section 2, for the case of *large but finite* ESS-values.

As to determine the most likely graph structure, note that the marginal likelihood $p(D|\alpha, G^+)$ of a graph G^+ is important only *relative* to the marginal likelihood $p(D|\alpha, G^-)$ of a competing graph G^-. In particular, we consider two graphs in the following that are identical except for the edge $A \leftarrow B$, which is present in G^+ and absent in G^-. Let Π denote the parents of A in graph G^-; this implies that, in G^+, the parents of A are Π and B. The presence of the edge $A \leftarrow B$ given the parents Π is favoured over its absence (i.e., graph G^+ over G^-) if the log Bayes factor $\log p(D|\alpha, G^+)/p(D|\alpha, G^-)$ is positive; and if negative, the absence of $A \leftarrow B$ is favoured. Given complete data (i.e., no missing data), the marginal likelihood factorizes (cf. Eq. 2), so that most terms in the Bayes factor cancel out, and it depends only on the variables A and B and the parents Π:

$$\begin{aligned}\log \frac{p(D|\alpha,G^+)}{p(D|\alpha,G^-)} &= \sum_{a,b,\pi} \log \frac{\Gamma(N_{a,b,\pi} + \alpha_{a,b,\pi})}{\Gamma(\alpha_{a,b,\pi})} \\ &- \sum_{a,\pi} \log \frac{\Gamma(N_{a,\pi} + \alpha_{a,\pi})}{\Gamma(\alpha_{a,\pi})} - \sum_{b,\pi} \log \frac{\Gamma(N_{b,\pi} + \alpha_{b,\pi})}{\Gamma(\alpha_{b,\pi})} \\ &+ \sum_{\pi} \log \frac{\Gamma(N_{\pi} + \alpha_{\pi})}{\Gamma(\alpha_{\pi})} \qquad (7)\end{aligned}$$

Note that this Bayes factor is symmetric in A and B, as expected for independence tests; the asymmetry of the edge $A \leftarrow B$ is caused by Π being the parents of A (rather than of B) in the graph. Now we can present our main result for large but finite ESS-values:

Proposition 2: *Given the BDeu score, the leading-order approximation of the Bayes factor of the two graphs G^+*

and G^- defined above reads for large ESS-values ($\alpha \gg N$):

$$\log \frac{p(D|\alpha, G^+)}{p(D|\alpha, G^-)} = \frac{N}{2\alpha} \left\{ N \cdot U(\hat{p}(A,B|\Pi)) - d_{\mathrm{F}} \right\} + \mathcal{O}(N^2/\alpha^2) \qquad (8)$$

where U is the uniformity measure as defined above, $\hat{p}(A,B|\Pi)$ is the empirical distribution implied by the data, i.e., $\hat{p}(a,b|\pi) = N_{a,b,\pi}/N_\pi$, and $d_{\mathrm{F}} = |\Pi|(|A|-1)(|B|-1)$ are the (well known) degrees of freedom.[1]

Before we give the proof at the end of this section, let us first discuss interesting insights that follow from this approximation:

First, note that if U were replaced by the (conditional) mutual information I, then the term $N \cdot I - d_{\mathrm{F}}$ would be identical to the Akaike Information Criteria (AIC) [Akaike, 1973]. This analogy suggests that, in Eq. 8, d_{F} serves as a penalty for model complexity. In other words, $N \cdot U(\hat{p}) - d_{\mathrm{F}} > 0$ means that $U(\hat{p})$ is *notably* (or significantly) larger than zero, while $N \cdot U(\hat{p}) - d_{\mathrm{F}} < 0$ refers to $U(\hat{p})$ being not notably larger than zero. Given the properties of the new measure U (as discussed in Section 3.1), it hence follows directly:

Corollary 1: *Given the BDeu score, there exists a value $\alpha^+ \in \mathbb{R}$ such that for all finite $\alpha > \alpha^+$ the presence of the edge $A \leftarrow B$ given the parents Π is favoured over its absence if $N \cdot U(\hat{p}(A,B|\Pi)) - d_{\mathrm{F}} > 0$, i.e., if the empirical distribution $\hat{p}$ implies*

- *a notable dependence between A and B given Π,*
- **or** *a notable skewness (i.e., non-uniformity) of both distributions $\hat{p}(A|\Pi)$ and $\hat{p}(B|\Pi)$. Note that A and B can be conditionally independent here.*

Conversely, the absence of the edge $A \leftarrow B$ given the parents Π is favoured for all large values $\alpha \gg N$ if $N \cdot U(\hat{p}(A,B|\Pi)) - d_{\mathrm{F}} < 0$, i.e., if the empirical distribution implies that

- *A and B are not notably dependent given Π,*
- **and** *for each state π with $p(\pi) > 0$ there is one marginal distribution $\hat{p}(A|\pi)$ or $\hat{p}(B|\pi)$ that is not notably different from a uniform distribution.*

Second, given that this applies to any edge in the graph, it follows immediately that the *complete* graph achieves the largest marginal likelihood if a sufficiently large (but finite) ESS-value is chosen, assuming that, for each variable conditioned on any set of parents, the data implies notable dependence or a sufficiently skewed distribution. Note, however, that the latter condition may not necessarily hold for large sets of parents when the given data set is *small*; this is because many zero-cell counts occur if the joint number of states of the variables is larger than the number of samples in the data.

Proof of Proposition 2: Each of the four terms in the Bayes factor in Eq. 7 takes the form $\sum_y \log(\Gamma(N_y + \alpha_y)/\Gamma(\alpha_y))$, where y denotes a (joint) state y of a set Y of random variables. If $\alpha_y \gg N_y$, we obtain for each y (using $\Gamma(z) = (z-1)!$ in the first line):

$$\begin{aligned} \log \frac{\Gamma(N_y + \alpha_y)}{\Gamma(\alpha_y)} &= \sum_{k=1}^{N_y} \log(k - 1 + \alpha_y) \\ &= \sum_{k=1}^{N_y} \log \alpha_y + \sum_{k=1}^{N_y} \log \frac{k-1+\alpha_y}{\alpha_y} \\ &= N_y \log \alpha_y + \sum_{k=1}^{N_y} \frac{k-1}{\alpha_y} + \mathcal{O}(N_y^2/\alpha_y^2) \\ &= N_y \log \alpha_y + \frac{N_y(N_y-1)}{2\alpha_y} + \mathcal{O}(N_y^2/\alpha_y^2). \end{aligned} \qquad (9)$$

Inserting this approximation into the log Bayes factor of Eq. 7, we obtain (note that $\alpha_{a,b,\pi} = \alpha/|A,B,\Pi|$ for the BDeu score, where $|\cdot|$ denotes the number of joint states of the random variables):

$$\begin{aligned} &\log \frac{p(D|G^+)}{p(D|G^-)} \\ &= \sum_{a,b,\pi} N_{a,b,\pi} \log \frac{\alpha_{a,b,\pi}\alpha_\pi}{\alpha_{b,\pi}\alpha_{a,\pi}} \\ &\quad + \frac{1}{2\alpha} \sum_{a,b,\pi} N_{a,b,\pi} \big(|A,B,\Pi| \cdot N_{a,b,\pi} - |A,\Pi| \cdot N_{a,\pi} \\ &\qquad\qquad - |B,\Pi| \cdot N_{b,\pi} + |\Pi| \cdot N_\pi \big) \\ &\quad - \frac{1}{2\alpha} \sum_{a,b,\pi} N_{a,b,\pi} \big(|A,B,\Pi| - |A,\Pi| - |B,\Pi| + |\Pi| \big) \\ &\quad + \mathcal{O}(N^2/\alpha^2) \end{aligned} \qquad (10)$$

where the first term vanishes because $\alpha_{A,B,\Pi}$ is uniform for the BDeu score, the second term equals $N^2 \cdot U(\hat{p}(A,B|\Pi))/(2\alpha)$,[2] and the third term equals $d_{\mathrm{F}} \cdot N/(2\alpha)$. □

3.3 ILLUSTRATION

This section provides a simple example that shows how the combination of a uniform (prior) distribution and a skewed empirical distribution can create dependence while each individual distribution implies independence. The key is

[1] Here, d_{F} must not be corrected for zero-cell counts, as it originates from the prior in the BDeu score, cf. Eq. 10. Note that this is different from d_G^{eff} in Section 4.2.

[2] Because of $N_{a,b,\pi} = N \cdot \hat{p}(a,b,\pi)$.

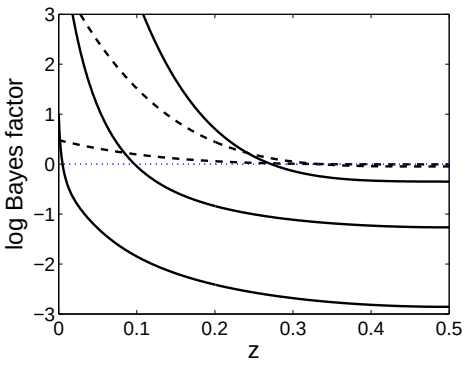


Figure 1: This example illustrates how the dependence increases with the skewness implied by the data for several ESS-values: 100, 10 , 1 (solid curves, from top to bottom); 1,000 and 10,000 (dashed curves, from top to bottom).

that, in this Bayesian approach with the Dirichlet prior, the log Bayes factor is essentially based on a weighted *sum* of two distributions (see Section 2 and Eq. 3): the empirical distribution $\hat{p}$ (with weight N) and the uniform distribution q of the 'virtual' data points due to the Dirichlet prior (with weight α). Obviously, the *sum* of two distributions (where each implies independence) does not necessarily result in a distribution that implies independence: $N\hat{p}(A)\hat{p}(B) + \alpha q(A)q(B) \neq c \cdot p(A)p(B)$ for any distribution p and any constant c in general. The deviation from this equality is typically not statistically significant for uniform q and skewed $\hat{p}$ as long as the ESS-value α is sufficiently small; but as the ESS-value α increases in this weighted sum, the deviation can become statistically significant, as illustrated in the following example: let us consider an artificial data set D with $N = 100$ samples implying *independence* between two binary variables A and B, i.e., the empirical distribution factorizes like $\hat{p}(A,B) = \hat{p}(A)\hat{p}(B)$; for simplicity, we assume that $\hat{p}(A) = \hat{p}(B)$. We control the skewness of the marginal distributions with a parameter $z \in [0, 0.5]$: $\hat{p}(A = 1) = z$ and $\hat{p}(A = 0) = 1 - z$; $z = 0.5$ implies uniform distribution, while the skewness of the distribution grows as z decreases. Values in the range $[0.5, 1]$ are symmetric to the case considered here. Figure 1 illustrates how the log Bayes factor (cf. Eq. 7) increases from negative values (implying independence) to positive values (suggesting dependence) as the empirical distribution becomes increasingly skewed (i.e., as z decreases). Figure 1 shows that, as the ESS-value increases, the degree of skewness of $\hat{p}$ has to diminish as to prevent the log Bayes factor from implying dependence. As an aside, note that the curves are quite flat for large and small ESS-value where either one of the distributions dominates the sum; in contrast, the maximum at $z = 0$ is reached for the ESS-value $\alpha = N = 100$, i.e., when both distributions are weighted equally.

4 UNDERSTANDING THE OPTIMAL ESS-VALUE

Having theoretically analyzed the cases of large ESS-values in the first part of this paper, and the case of small ESS-values in [Steck & Jaakkola, 2002], the remainder of this paper focuses on the 'optimal' ESS-value (in a predictive sense): we tackle the question as to why the 'optimal' ESS-value is about 50 for some of the 20 UCI data sets in the elaborate experiments in [Silander *et al.*, 2007] (their results are also reproduced in columns α^{M} and α^{I} in our Table 1 for comparison), while it ranges between 2 and 13 for the remaining data sets. This section aims to provide an answer to this question, i.e., the goal is to *understand* as to which properties of the data determine the 'optimal' ESS-value.

4.1 OPTIMIZATION OF GRAPH AND ESS

Given that the ESS-value can have a crucial effect on the learned network structure [Silander *et al.*, 2007, Steck & Jaakkola, 2002], we now depart from the orthodox Bayesian approach of choosing the prior—including the ESS-value— without having seen the data. In the remainder of this paper we treat the ESS α as a latent random variable to be learned in light of the data D. Various objective functions for determining the 'optimal' ESS value α were discussed in [Silander *et al.*, 2007], and found to yield essentially the same result. In this paper, we choose to jointly optimize the graph structure and the ESS-value:

$$(\alpha^*, G^*) = \arg\max_{(\alpha,G)} p(D|\alpha, G) \tag{11}$$

While joint optimization of Eq. 11 is computationally very expensive, a simple coordinate-wise ascent is more tractable. This approach comprises two alternate update-steps in each round $k = 0, 1, \ldots$ (until convergence):

1. optimize the graph for a fixed ESS-value: $G^*_k = \arg\max_G p(D|\alpha^*_{k-1}, G)$,
2. optimize the ESS-value for a fixed graph: $\alpha^*_k = \arg\max_\alpha p(D|\alpha, G^*_k)$.

The first step can be solved by any standard algorithm for structure learning (cf. the review in Section 2). Concerning the initialization of the graph G_0 in this iterative algorithm, there are several options. A convenient choice is to learn the graph G_0 that optimizes the Bayesian Information Criteria (BIC) [Schwarz, 1978, Haughton, 1988]. This criteria not only is a large-sample approximation to the log marginal likelihood in Eq. 2, but it also is independent of ESS, i.e., it does not contain a free parameter. It can hence be used for initialization of G_0 when no initial ESS-value is known, and promises to yield an initial graph G_0 that is

already close to the optimum with respect to the marginal likelihood.

The main contribution of the remainder of this paper is to derive an analytical approximation for the optimal α^* in the second update-step.

4.2 OPTIMAL ESS FOR GIVEN GRAPH

This section provides an understanding of the most important properties of the data that affect the value of the optimal ESS. We derive an analytical approximation to the optimal ESS-value for a fixed graph G, cf. step 2 above.

Step 2 maximizes predictive accuracy in the prequential sense [Heckerman *et al.*, 1995, Dawid, 1984], when the data points are considered to arrive individually in a sequence. As this optimization problem is difficult to solve, we now replace it by a similar, but *frequentist* objective, departing from *Bayesian* statistics: we minimize the *test error*, as is commonly done in cross-validation. Even though this objective is not the same as the original one, it can be expected to yield a sufficiently accurate approximation in as far as we only aim to understand the difference between ESS-values of about 50 versus about 10 or lower for the various data sets in [Silander *et al.*, 2007]. Note that this assumption and the following ones are validated experimentally on 20 UCI data sets in Section 4.3.

In the following, we combine two approximations to the test error as to obtain an explicit approximation to the optimal ESS-value α^*. We carry out both approximations w.r.t. a 'reference point' in the space of distributions. For convenience, we choose this to be the distribution $\hat{p}(X|G)$ implied by the Bayesian network model with graph G and maximum-likelihood parameter estimates $\hat{\theta}$. The test error of this model with respect to the true distribution $p(X)$ reads

$$T[p(X), \hat{p}(X|G)] = -\sum_x p(x) \log \hat{p}(x|G), \qquad (12)$$

when using the log loss. As p is unknown, this cannot be evaluated. As is well-known [Akaike, 1973], however, the test error can be approximated by the *training error*, $-E_{\hat{p}(X)}[\log \hat{p}(X|G)]$, and a penalty term for model complexity:

$$T[p(X), \hat{p}(X|G)] = -E_{\hat{p}(X)}[\log \hat{p}(X|G)] + \frac{d_G^{\text{eff}}}{N} + \mathcal{O}(\frac{1}{N^2}), \qquad (13)$$

where

$$\begin{aligned} E_{\hat{p}(X)}[\log \hat{p}(X|G)] &= \sum_x \hat{p}(x) \log \hat{p}(x|G) \\ &= \sum_{i=1}^{n} \sum_{x_i} \sum_{\pi_i} \frac{N_{x_i,\pi_i}}{N} \log \frac{N_{x_i,\pi_i}}{N_{\pi_i}}, \qquad (14) \end{aligned}$$

equals the maximum log likelihood divided by the sample size, and d_G^{eff} is the *effective number of parameters* of the Bayesian network model:

$$d_G^{\text{eff}} = \sum_{i=1}^{n} \left[\sum_{x_i,\pi_i} I(N_{x_i,\pi_i}) - \sum_{\pi_i} I(N_{\pi_i}) \right], \qquad (15)$$

where $I(\cdot)$ is the indicator function: $I(a) = 1$ if $a > 0$ and $I(a) = 0$ otherwise. If the data set is sufficiently large so that all cell counts N_{x_i,π_i} are positive, then d_G^{eff} equals the well-known number of parameters, which is given for Bayesian networks by $\sum_i (|X_i| - 1)|\Pi_i|$, where $|\cdot|$ denotes the number of (joint) states of the variable(s).

We obtain the second approximation to the test error in Eq. 12 as follows: we assume the (unknown) true distribution p is well-approximated by the functional form of the regularized parameter estimates in Eq. 3 using the optimal value α^*:

$$p_{x_i,\pi_i} \approx \tilde{p}_{x_i,\pi_i}, \qquad (16)$$

where

$$\begin{aligned} \tilde{p}_{x_i,\pi_i} &= \frac{N_{x_i,\pi_i} + \alpha^* \cdot q_{x_i,\pi_i}}{N + \alpha^*} \\ &= \hat{p}(x_i, \pi_i) + \frac{\alpha^*}{N} \left(q_{x_i,\pi_i} - \frac{N_{x_i,\pi_i}}{N} \right) + \mathcal{O}((\frac{\alpha^*}{N})^2). \qquad (17) \end{aligned}$$

This first-order Taylor expansion about the maximum-likelihood estimate (our 'reference point') assumes $\alpha^* \ll N$, which will be justified at the end of this section. Inserting Eqs. 16 and 17 into Eq. 12, now results in our second approximation,

$$\begin{aligned} T[p(X), \hat{p}(X|G)] \approx & -E_{\hat{p}(X)}[\log \hat{p}(X|G)] \\ & -\frac{\alpha^*}{N} \sum_x (q_x - \frac{N_x}{N}) \log \hat{p}(x|G) + \mathcal{O}\left((\frac{\alpha^*}{N})^2 \right) \end{aligned}$$

Next we equate this approximation with the one in Eq. 13, and finally arrive at an explicit approximation to the optimal ESS-value,

$$\alpha^* \approx \frac{d_G^{\text{eff}}}{E_{\hat{p}(X)}[\log \hat{p}(X|G)] - E_{q(X)}[\log \hat{p}(X|G)]} + \mathcal{O}(\frac{\alpha^{*2}}{N}), \qquad (18)$$

where $E_{q(X)}[\log \hat{p}(X|G)]$ is the expectation with respect to the prior distribution q, analogous to Eq. 14; regarding the robustness against zero cell counts, we use $N^+_{x_i,\pi_i} = \max\{N_{x_i,\pi_i}, 1\}$ in place of N_{x_i,π_i} in practice.

Note that the denominator in Eq. 18 is indeed positive if the prior distribution q has a larger entropy H than the empirical distribution $\hat{p}$ does, which is the case for the BDeu score. This is obvious from

$$\begin{aligned} & E_{\hat{p}(X)}[\log \hat{p}(X|G)] - E_{q(X)}[\log \hat{p}(X|G)] \\ & = H(q(X|G)) - H(\hat{p}(X|G)) + \text{KL}(q(X|G)||\hat{p}(X|G)) \end{aligned}$$

given that the Kullback-Leibler divergence and the entropy H are non-negative.

Interpretation of Eq. 18: This explicit approximation of the optimal ESS α^* provides an understanding of the main properties of the data determining the optimal ESS-value:

- *skewness and dependence*: the denominator of Eq. 18 tends to increase when the entropy (or negative likelihood) of the empirical distribution factored according to the graph structure, $\hat{p}(X|G)$, decreases: this is the cases if there are strong dependencies along the edges, or if the conditional distributions of the variables are very skewed. For such data sets, one can hence expect small values of α^*. Conversely, data sets that do neither imply a skewed distribution nor extremely strong dependencies will result in increased values of α^*.
- *sample size*: Eq. 18 does not explicitly depend on the sample size N. There is an indirect relation, however: the effective number of parameters, which is a measure of model complexity, tends to increase with N; a more complex optimal model also entails an increased maximum likelihood, and hence this also affects the denominator; one may hence expect both the enumerator and the denominator to grow in a similar way as N increases, which suggests a weak dependence of α^* on N. Apart from that, when N is sufficiently large, the model complexity tends to approach its 'true' value and not increase further. Thus, for a sufficiently large data set, α^* becomes independent of N. Consequently, our earlier assumption $\alpha^* \ll N$ indeed holds for sufficiently large N. Note that this behavior is also consistent with the one expected in the asymptotic limit ($N \to \infty$), namely $\alpha^*/N \to 0$, so that the effect of regularization vanishes.
- *number of nodes*: Eq. 18 implies that α^* can be expected to be unaffected by the number n of nodes in the (sparse) graph on average because both the enumerator and denominator are additive w.r.t. the nodes, and hence on average grow proportionally to each other when adding additional nodes to the network.

4.3 EXPERIMENTS

As an additional confirmation of our assumptions underlying the approximations in Section 4.2, we determined the optimal value α^* based on our iterative algorithm using Eq. 18 on 20 UCI data sets [Hettich & Bay, 1999]. We used the same pre-processed data as was used in [Silander *et al.*, 2007] (imputation of missing values, discretization of continuous variables), and compared to their exact results.

Table 1 summarizes the results, and confirms the validity of our approximation. It is obvious that our approximate α^* agrees very well with the exact results of [Silander *et al.*, 2007], which were obtained under heavy computational costs there (in our Table 1, α^{M} is the exact solution of the maximization problem in Eq. 11, and α^I is the range of ESS-values that all yield the same MAP graph as α^{M} does [Silander *et al.*, 2007]): while our approximation α^* does not always agree precisely with the exact value α^{M}, it correctly identifies the data sets where the optimal ESS is about 50 as opposed to about 10 or lower for the remaining data sets. This suggests that our analytical approximation, and the underlying assumptions, capture the main effects that influence the optimal ESS-value. Moreover, note that the sample size N varies between about 100 and 60,000, and the number of variables n ranges from 5 to 15. Table 1 shows that both N and n have no obvious effect on the optimal ESS-value, as expected from our analytical approximation (see discussion in Section 4.2).

Table 1: Experimental validation of our analytical approximations in Section 4.2. See text for details.

Data	N	n	α^I	α^{M}	α^*	k
balance	625	5	1...100	48	44	1
iris	150	5	1...3	2	2	2
thyroid	215	6	2...2	2	3	5
liver	345	7	3...6	4	3	3
ecoli	336	8	7...10	8	8	3
abalone	4177	9	6...6	6	7	3
diabetes	768	9	3...5	4	3	3
post op	90	9	3...5	3	3	3
yeast	1484	9	1...6	6	6	2
cancer	286	10	6...10	8	7	2
shuttle	58000	10	1...3	3	3	2
tictac	958	10	51...62	51	60	2
bc wisc	699	11	7...15	8	5	3
glass	214	11	5...6	6	6	4
page	5473	11	3...3	3	3	2
heart cl	303	14	13...16	13	9	3
heart hu	294	14	5...6	5	5	3
heart st	270	14	7...10	10	10	4
wine	178	14	8...8	8	7	3
adult	32561	15	48...58	50	49	3

Moreover, note that the results of our two contributions are very similar to each other (see Sections 3 and 4): the decisive properties of the data are the implied dependencies, or—in case of independence—the skewness of the implied distribution.

With this insight, it is not surprising that an increase in the optimal ESS value is related to a reduction in the maximum number of edges in the graph attainable in the experiments of [Silander *et al.*, 2007]: the data sets with a *large* optimal ESS-value of about 50 are exactly the data sets for which the maximum number of edges in the graph (achieved by increasing the ESS-value) is less than 80% of the edge-count of the complete graph (see the column 'range' in Ta-

ble 1 of [Silander *et al.*, 2007]). Both have the same cause, namely a data set that implies neither strong dependencies nor skewness.

As an aside, note that our iterative algorithm (cf. Section 4.1 using the approximation in Eq. 18) is computationally efficient, as it converges within a small number of iterations, cf. k in Table 1; as a convergence criteria we required $|\alpha^*_k - \alpha^*_{k-1}| < 0.1$.

5 CONCLUSIONS

This paper presents two contributions that shed light on how the learned graph is affected by the value of the equivalent sample size (ESS). First, we analyzed theoretically the case of *large but finite* ESS values, which complements the results for small values in the literature. Among other results, it was surprising to find that the presence of an edge in a Bayesian network is favoured over its absence even if both the Dirichlet prior and the data imply independence, as long as the conditional empirical distribution is notably different from uniform. Our second contribution provides an understanding of which properties of the given data determine the optimal ESS value in a predictive sense (when considered as a free parameter). Our analytical approximation (which we also validated experimentally) shows that the optimal ESS-value is approximately independent of the number of variables in the (sparse) graph and of the sample size. Moreover, the optimal ESS-value is small if the data implies a skewed distribution or strong dependencies along the edges of the graph. Interestingly, this condition concerning the optimal ESS-value is very similar to the one derived for large but finite ESS-values. Finally, having shown the crucial effect of the ESS value on the graph structure that *maximizes* the Bayesian score, this suggests that a similar effect can be expected concerning the posterior *distribution* over the graphs, and hence for Bayesian model *averaging*. If one embarks on this popular Bayesian approach, one hence has to choose the prior with great care.

Acknowledgements

I am grateful to R. Bharat Rao for encouragement and support of this work, to Tomi Silander and Petri Myllymäki for providing me with their pre-processed data, and to the anonymous reviewers for valuable comments.

References

[Akaike, 1973] Akaike, H. 1973. Information Theory and an extension of the maximum likelihood principle. *Pages 267–81 of:* Petrox, B. N., & Caski, F. (eds), *Second International Symposium on Information Theory*.

[Buntine, 1991] Buntine, W. 1991. Theory refinement on Bayesian networks. *Pages 52–60 of:* D'Ambrosio, B., Smets, P., & Bonissone, P. (eds), *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann.

[Chickering, 1996] Chickering, D. M. 1996. Learning Bayesian networks is NP-complete. *Pages 121–30 of: Proceedings of the International Workshop on Artificial Intelligence and Statistics*.

[Cowell *et al.*, 1999] Cowell, R. G., Dawid, A. P., Lauritzen, S. L., & Spiegelhalter, D. J. 1999. *Probabilistic Networks and Expert Systems*. Springer.

[Dawid, 1984] Dawid, A. P. 1984. Statistical Theory. The prequential approach. *Journal of the Royal Statistical Society, Series A*, **147**, 277–305.

[Haughton, 1988] Haughton, D. M. A. 1988. On the choice of a model to fit data from an exponential family. *The Annals of Statistics*, **16**(1), 342–55.

[Heckerman *et al.*, 1995] Heckerman, D., Geiger, D., & Chickering, D. M. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, **20**, 197–243.

[Hettich & Bay, 1999] Hettich, S., & Bay, D. 1999. *The UCI KDD archive*. http://kdd.ics.uci.edu.

[Koivisto & Sood, 2004] Koivisto, M., & Sood, K. 2004. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, **5**, 549–73.

[Rissanen, 1978] Rissanen, J. 1978. Modeling by shortest data description. *Automatica*, **14**, 465–71.

[Schwarz, 1978] Schwarz, G. 1978. Estimating the dimension of a model. *The Annals of Statistics*, **6**(2), 461–64.

[Silander & Myllymäki, 2006] Silander, T., & Myllymäki, P. 2006. A Simple Approach for finding the globally optimal Bayesian network structure. *Pages 445–52 of: Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

[Silander *et al.*, 2007] Silander, T., Kontkanen, P., & Myllymäki, P. 2007. On Sensitivity of the MAP Bayesian Network Structure to the Equivalent Sample Size Parameter. *In: Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

[Steck & Jaakkola, 2002] Steck, H., & Jaakkola, T. S. 2002. On the Dirichlet Prior and Bayesian Regularization. *In: Advanves in Neural Information Processing Systems (NIPS) 15*.

[Tsallis, 2000] Tsallis, C. 2000. *Entropic Nonextensivity: a possible measure of complexity*. arXiv:cond-mat/0010150v1.

New Techniques for Algorithm Portfolio Design

Matthew Streeter and **Stephen F. Smith**
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
{matts, sfs}@cs.cmu.edu

Abstract

We present and evaluate new techniques for designing algorithm portfolios. In our view, the problem has both a scheduling aspect and a machine learning aspect. Prior work has largely addressed one of the two aspects in isolation. Building on recent work on the scheduling aspect of the problem, we present a technique that addresses both aspects simultaneously and has attractive theoretical guarantees. Experimentally, we show that this technique can be used to improve the performance of state-of-the-art algorithms for Boolean satisfiability, zero-one integer programming, and A.I. planning.

1 Introduction

Many computational problems that arise in the world are NP-hard, and thus likely to be intractable from a worst-case point of view. However, the particular instances of these problems that are actually encountered can often be solved effectively using heuristics that do not have good worst-case guarantees. Typically there are a number of heuristics available for solving any particular NP-hard problem, and there is no one heuristic that performs best on all problem instances. Thus, when solving a particular instance of an NP-hard problem, it is not clear *a priori* how to best make use of the available CPU time.

Specifically, suppose you wish to solve an instance x of a computational problem, and there are k heuristics available for solving it. Each heuristic, when run on instance x, will either solve the instance in finite time (e.g., by returning a provably correct "yes" or "no" answer to a decision problem, returning a provably optimal solution to an optimization problem), or will run forever without solving it. When solving x, you will in general have some prior knowledge of how each of the k heuristics behaves on other instances of the same computational problem. Naturally, you would like to solve x as quickly as possible.

In this situation, a natural approach would be to label each previously-encountered problem instance with a set of features, and then to use some machine learning algorithm to predict which of the k heuristics will return an answer in the shortest amount of time. However, if we then run the predicted fastest heuristic and it does not yield an answer after some sufficiently large amount of time, we might suspect that the machine learning algorithm's prediction was a mistake, and might try running a different heuristic instead. Alternatively, if the heuristic is randomized, we might try restarting it and running with a fresh random seed.

We refer to the general problem of determining how to solve a problem instance in this setting as *algorithm portfolio design* [5, 6]. As just illustrated, the problem has both a machine learning aspect (predicting which heuristic will solve the instance first) and a scheduling aspect (determining how long to run a heuristic before giving up and trying a different heuristic). Previous work (e.g., [6, 8, 10]) has largely addressed one of the two aspects in isolation (we discuss previous work in detail in §6). In this work, we present an approach that addresses both aspects of the problem simultaneously and has attractive theoretical guarantees.

We note up front that our work does not address all possible aspects of the algorithm portfolio design problem. For example, we ignore the possibility of making scheduling decisions dynamically based on the observed behavior of the heuristics (e.g., if a heuristic has a progress bar that indicates how close it is to solving the instance). We also ignore the possibility of sharing information (e.g., upper and lower bounds on the optimal value of the objective function) between heuristics as they are executing.

1.1 Formal setup

We are given as input a set $\mathcal{H}$ of heuristics (i.e., algorithms with potentially large running time) for solving some computational problem. Heuristic h, when run on problem instance x, runs for $T(h,x)$ time units before solving the problem. If h is randomized, then $T(h,x)$ is a random variable whose outcome depends on the sequence of random bits supplied as input to h.

We will be interested in interleaving the execution of heuristics according to *schedules* of the following form.

Definition (schedule). *A* schedule $S = \langle (h_1,\tau_1),(h_2,\tau_2),\ldots\rangle$ *is a sequence of pairs* $(h,\tau) \in \mathcal{H} \times \mathbb{R}_{>0}$*, where each pair* (h,τ) *represents running heuristic* h *for time* t*.*

When interpreting a schedule, we allow each heuristic $h \in \mathcal{H}$ to be executed in one of two models (the choice of model need not be the same for all heuristics). If h is executed in the *suspend-and-resume model*, then a pair (h,τ) represents continuing a run of heuristic h for an additional τ time units. The run of h is then temporarily suspended and kept resident in memory, to be potentially resumed later on. In contrast, if h is executed in the restart model, then a pair (h,τ) represents running h from scratch for time τ, and then deleting the run from memory (if h is randomized, the run is performed with a fresh random seed).

Abusing notation slightly, we use $T(S,x)$ to denote the time required to solve problem instance x using schedule S. We illustrate the definition of $T(S,x)$ with an example. Consider the schedule

$$S = \langle (h_1,2),(h_2,2),(h_1,4),\ldots\rangle$$

illustrated in Figure 1. Suppose $\mathcal{H} = \{h_1,h_2\}$, both heuristics are deterministic, and $T(h_1,x) = T(h_2,x) = 3$. Then $T(S,x) = 5$ if h_1 is executed in the suspend-and-resume model, whereas $T(S,x) = 7$ if h_1 is executed in the restart model. Note that in calculating $T(S,x)$ when S is executed in the suspend-and-resume model, we ignore any overhead associated with context-switching.

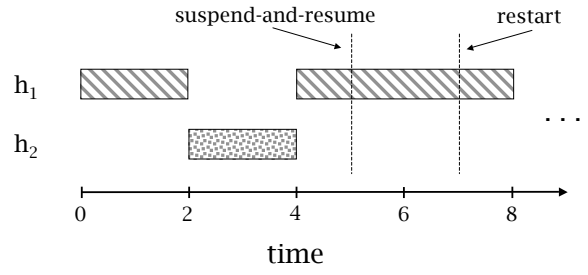


Figure 1: Value of $T(S,x)$ in two execution models.

This class of schedules is quite flexible, and includes *restart-schedules* [11] and *task-switching schedules* [13] as special cases. A restart schedule is a schedule for a single randomized heuristic, executed in the restart model. A task-switching schedule is a schedule for a set of one or more deterministic heuristics, each executed in the suspend-and-resume model.

An *algorithm portfolio* is a way to decide what schedule to use to solve a particular problem instance.

Definition (algorithm portfolio). *An* algorithm portfolio *is a procedure* ϕ *that, given a problem instance* x*, returns a schedule* $\phi(x)$ *to use to solve* x*.*

We measure the performance of a schedule S on a problem instance x in terms of $\mathbb{E}[T(S,x)]$, where the expectation is over the random bits used in the runs that S performs. We are interested in optimizing this objective in two settings: offline and online.

In the offline setting, we are given as input a set of training instances, along with the value of $T(x,h)$ (or in general, an estimate of its distribution) for each heuristic h and training instance x. Our goal is to construct an algorithm portfolio (within some class) that performs optimally on the set of training instances. We would then use such a portfolio to solve additional, similar problem instances more efficiently.

In the online setting, we are fed a sequence $\mathcal{X} = \langle x_1,x_2,\ldots,x_n\rangle$ of problem instances one at a time and must obtain a solution to each instance (via some schedule) before moving on to the next instance. When selecting a schedule S_i to use to solve instance x_i, we have knowledge of the previous instances $x_1,x_2,\ldots,x_{i-1}$ but we have no knowledge of x_i itself or of any subsequent instances. In this setting, our goal is to learn an effective algorithm portfolio on-the-fly, again with the aim of minimizing average CPU time.

1.2 Summary of results

In §2, we review recent results on a pure scheduling approach to the algorithm portfolio design problem. For the offline setting, the main result is a greedy algorithm that returns a 4-approximation to the optimal schedule; achieving a $4-\epsilon$ approximation for any $\epsilon > 0$ is NP-hard. For the online setting, the main result is an online schedule-selection algorithm whose worst-case performance guarantees converge to those of the offline greedy approximation algorithm, asymptotically as the number of instances grows large. Note that the latter guarantee does not require any statistical assumptions about the sequence of problem instances.

In §3, we discuss how the online algorithm discussed in §2 can be combined with algorithms for solving the

so-called *sleeping experts problem* in order to take advantage of Boolean features of an instance when selecting a schedule. This approach yields an online algorithm that, simultaneously for each feature f, is guaranteed to perform near-optimally (i.e., average CPU time asymptotically at most 4 times that of any schedule) on the subset of instances for which f is true.

In §4, we evaluate these techniques experimentally, and show that they can be used to improve the performance of state-of-the-art heuristics for Boolean satisfiability, A.I. planning, and zero-one integer programming.

The results just described apply only to the objective of minimizing average CPU time. In §5, we consider the case in which each heuristic is an *anytime algorithm* that returns solutions of increasing quality over time. We describe how our results for minimizing average CPU time can be generalized to yield schedules with good anytime behavior, and demonstrate the power of this approach by applying it to state-of-the-art heuristics for zero-one integer programming.

2 Background

In this section we review recent results on a pure scheduling approach to algorithm portfolio design. These results form the basis of the algorithms and experimental results presented in the rest of the paper.

2.1 Offline greedy approximation algorithm

Suppose we collect a set of training instances $\mathcal{X}$, and wish to compute the schedule that performs optimally over the training instances (i.e., the schedule S that minimizes $\sum_{x\in\mathcal{X}} \mathbb{E}\left[T\left(S,x\right)\right]$). We assume that for each heuristic $h \in \mathcal{H}$ and training instance $x \in \mathcal{X}$, the distribution of $T\left(h,x\right)$ is known exactly (in practice, we would have to estimate it by performing a finite number of runs).

Building on previous work on the Min-Sum Set Cover problem [3], Streeter *et al.* [16, 17] developed a greedy approximation algorithm for this offline problem. Let $f(S)$ denote the sum, over all instances $x \in \mathcal{X}$, of the probability that executing schedule S yields a solution to instance x. The schedule $G = \langle g_1, g_2, \ldots\rangle$ returned by the greedy approximation algorithm can be defined inductively as follows: $G_1 = \langle\rangle$, $G_j = \langle g_1, g_2, \ldots, g_{j-1}\rangle$ for $j > 1$, and

$$g_j = \operatorname*{arg\,max}_{a=(h,\tau)\in\mathcal{H}\times\mathbb{R}_{>0}} \left\{ \frac{f(G_j + a) - f(G_j)}{\tau} \right\} \qquad (1)$$

where $G_j + a$ denotes the schedule obtained by appending the pair a to G_j.[1] Informally, G is constructed by greedily appending a run $a = (h,\tau)$ to the schedule so as to maximize the expected number of instances a solves per unit time.

The performance of G is summarized by the following theorem. The theorem shows that, assuming P $\neq$ NP, the greedy schedule has optimal worst-case performance from an approximation standpoint (among schedules that can be computed in polynomial time).

Theorem 1 (Streeter *et al.*, 2007a; 2007b)**.** *G is a 4-approximation to the optimal schedule. That is,*

$$\sum_{x\in\mathcal{X}} \mathbb{E}\left[T\left(G,x\right)\right] \leq 4 \cdot \min_{S} \left\{ \sum_{x\in\mathcal{X}} \mathbb{E}\left[T\left(S,x\right)\right] \right\} .$$

Furthermore, for any $\epsilon > 0$, obtaining a $4-\epsilon$ approximation to the optimal schedule is NP-hard (even in the special case where all heuristics are deterministic).

2.2 Online greedy algorithm

In the online setting, a sequence $\langle x_1, x_2, \ldots, x_n\rangle$ of problem instances arrive one at a time, and one must solve each instance x_i via some schedule (call it S_i) before moving on to instance x_{i+1}. When selecting S_i, one has no knowledge of x_i itself. After solving x_i, one learns only the outcomes of the runs that were actually performed when executing S_i. As in the offline setting, the goal is to minimize the average CPU time required to solve each instance in the sequence.

Recently, Streeter and Golovin [15] developed an online algorithm for an abstract scheduling problem that includes this online problem as a special case. For the results of [15] to apply, we must make some additional assumptions. First, we assume that $T\left(h,x_i\right)$ is an integer for all heuristics h and instances x_i. Second, we assume that the CPU time the online algorithm uses up on any particular instance x_i is artificially capped at some value B (without such a cap, the online algorithm could be forced to spend an arbitrarily large amount of CPU time solving a single instance, and we could prove no meaningful bounds on its performance).

The algorithm presented in [15] is called **OG**, for "online greedy", and can be viewed as an online version of the greedy approximation algorithm described in §2.1. The following theorem shows that its worst-case performance guarantees approach those of the offline greedy algorithm, asymptotically as the number of problem instances approaches infinity. The theorem can be proved as a corollary of [15, Theorem 11] (for a formal derivation, see [14, Chapter 3]).

[1] Evaluating the arg max in (1) requires considering $O\left(r\left|\mathcal{X}\right|\right)$ values of τ per heuristic, where r is the maximum number of runs used to estimate the distribution of $T\left(h,x\right)$. For more details, see [14].

Theorem 2 (Streeter and Golovin, 2007)**.** *Algorithm* **OG** *[15], run with exploration probability* $\gamma = \Theta\left(n^{-\frac{1}{4}}\right)$*, has the following guarantee. Let* $T_i = \min\{B, T(S_i, x_i)\}$*, for some* $B > 0$*. Then*

$$\sum_{i=1}^{n} \mathbb{E}[T_i] \leq 4 \cdot \min_{S \in \mathcal{S}} \left\{ \sum_{i=1}^{n} \mathbb{E}[T(S, x)] \right\} + O\left(n^{\frac{3}{4}}\right) .$$

3 Exploiting Features

The algorithms referred to in theorems 1 and 2 provide no mechanism for tailoring the choice of schedule to the particular problem instance being solved. In practice, there may be quickly-computable features that distinguish one instance from another and suggest the use of different heuristics. In this section, we describe how existing techniques for solving the so-called *sleeping experts problem* can be used to exploit such features in an attractive way.

The sleeping experts problem is defined as follows. One has access to a set of M experts. On each day, a given expert is either *awake*, in which case the expert dispenses a piece of advice, or the expert is *asleep*. At the beginning of day i, one must select an awake expert whose advice to follow. Following the advice of expert j on day i incurs a loss $\ell_j^i \in [0, 1]$. At the end of day i, the value of the loss ℓ_j^i for each (awake) expert j is made public, and can be used as the basis for making choices on subsequent days. Note that the historical performance of an expert does not imply any guarantees about its future performance. Remarkably, randomized expert-selection algorithms nevertheless exist that achieve the following guarantee: simultaneously for each j, one's expected loss on the subset D_j of days when j was awake is at most $\sum_{i \in D_j} \ell_j^i + O\left(\sqrt{n \log M} + \log M\right)$. Thus, when using such an algorithm[2], one asymptotically performs as well as any fixed expert on the subset of days that expert was awake.

Suppose that each problem instance x_i is labeled with the values of M Boolean features. We will exploit such features by applying the sleeping experts algorithm in a standard way. We create, for each feature j, a copy $\mathcal{A}_j$ of the online schedule-selection algorithm **OG** that is only run on instances where feature j is true. We then use an algorithm for the sleeping experts problem to select among the schedules returned by the various copies, as described in the pseudo-code for $\mathbf{OG}^{se}$. Due to space constraints, the pseudo-code refers to [15, 17] for the details of certain steps. As in §2.2, we use B to denote an artificial bound on CPU time.

[2]See [2] for a description of such an algorithm. The algorithm maintains, for each expert, a weight that is adjusted based on its performance relative to other experts. On each day, experts are selected with probability proportional to their weights.

Algorithm $\mathbf{OG}^{se}$

Initialization: let $\mathcal{E}$ be a copy of the sleeping experts algorithm of [2]; and for each feature j, let $\mathcal{A}_j$ be a copy of **OG** [15].

For i from 1 to n:

1. Let F_i be the set of features that are true for x_i. For each feature $j \in F_i$, use $\mathcal{A}_j$ to select a schedule $S_{i,j}$.
2. Use $\mathcal{E}$ to select a feature (expert) $j_i \in F_i$, and select the schedule $S_i = S_{i,j_i}$.
3. With probability $\gamma = \Theta\left(n^{-\frac{1}{4}}\right)$, *explore* as follows. Using the procedure of [17], run each heuristic for time $O(B \log B)$ in order to obtain a function $\hat{f}$ such that for any schedule S, $\mathbb{E}\left[\hat{f}(S)\right] = \mathbb{E}[\min\{B, T(S, x_i)\}]$. Feed $\hat{f}$ back to each $\mathcal{A}_j$, as described in [15]. Finally, for each j, set $\ell_j^i = \frac{1}{B}\hat{f}(S_{i,j})$. Otherwise (with probability $1 - \gamma$) set $\ell_j^i = 0$ for all j.
4. For each $j \in F_i$, feed back ℓ_j^i to $\mathcal{E}$ as the loss for expert j.

The performance of $\mathbf{OG}^{se}$ is summarized by the following theorem.

Theorem 3. *Let* $\mathcal{X}_j$ *be the subset of instances for which feature* j *is true. Let* $T(x)$ *be the CPU time spent by* $\mathbf{OG}^{se}$ *on instance* x*. Then, simultaneously for each* j*, we have*

$$\mathbb{E}\left[\sum_{x \in \mathcal{X}_j} T(x)\right] \leq 4 \cdot \min_{S} \left\{ \sum_{x \in \mathcal{X}_j} \mathbb{E}[T(S, x)] \right\} + O\left(n^{\frac{3}{4}}\right) .$$

Proof. As already discussed, the algorithm $\mathcal{E}$ used as a subroutine in $\mathbf{OG}^{se}$ guarantees that, for any j,

$$\sum_{x \in \mathcal{X}_j} \ell_{j_i}^i \leq \sum_{x \in \mathcal{X}_j} \ell_j^i + R \tag{2}$$

where $R = O\left(\sqrt{n \log M} + \log M\right)$. Define $L_i(S) = \mathbb{E}[\min\{B, T(S, x_i)\}]$. Thus $\mathbb{E}\left[\ell_j^i\right] = \frac{\gamma}{B} L_i(S_{i,j})$. Taking the expectation of both sides of (2) yields

$$\sum_{x \in \mathcal{X}_j} L_i(S_i) \leq \sum_{x \in \mathcal{X}_j} L_i(S_{i,j}) + \frac{B}{\gamma} R .$$

Note that $\frac{B}{\gamma}R = O\left(n^{\frac{3}{4}}\right)$ (for constant M). At the same time, by Theorem 2 we have

$$\sum_{x\in\mathcal{X}_j} L_i(S_{i,j}) \leq 4 \cdot \min_S \left\{ \sum_{x\in\mathcal{X}_j} \mathbb{E}\left[T\left(S,x\right)\right] \right\} + O\left(n^{\frac{3}{4}}\right) .$$

Finally, because $\gamma = \Theta\left(n^{-\frac{1}{4}}\right)$, we have $\mathbb{E}\left[\sum_{x\in\mathcal{X}_j} T(x)\right] \leq \sum_{x\in\mathcal{X}_j} L_i(S_i) + O\left(n^{\frac{3}{4}}\right)$. Putting these equations together proves the theorem. □

Note that Theorem 3 provides a very strong guarantee. For example, if each instance is labeled as either "large" or "small" and also as either "random" or "structured", then the performance of $\mathbf{OG}^{se}$ on large instances will be nearly as good as that of the optimal schedule for large instances, and *simultaneously* its performance on structured instances will be nearly as good as that of the optimal schedule for structured instances (even though these subsets of instances overlap, and the optimal schedule for each subset may be quite different).

4 Experimental Evaluation

In this section, we evaluate the algorithms presented in the previous section experimentally using data from recent solver competitions.

4.1 Solver competitions

Each year, various computer science conferences hold competitions designed to assess the state of the art solvers in some problem domain. In these competitions, each submitted solver is run on a sequence of problem instances, subject to some per-instance time limit. Solvers are awarded points based on the instances they solve and how fast they solve them, and prizes are awarded to the highest-scoring solvers.

The experiments reported here make use of data from the following three solver competitions.

1. **SAT 2007**. Boolean satisfiability is the task of determining whether there exists an assignment of truth values to a set of Boolean variables that satisfies each clause (disjunction) in set of clauses. SAT solvers are used as subroutines in state-of-the-art algorithms for hardware and software verification and A.I. planning. The SAT 2007 competition included industrial, random, and hand-crafted benchmarks.

2. **IPC-5**. A.I. planning is the problem of finding a sequence of actions (called a plan) that leads from a starting state to a desired goal state, according to some formal model of how actions affect the state of the world. We used data from the *optimal planning* track of the Fifth International Planning Competition (IPC-5), in which the model of the world is specified in the STRIPS language and the goal is to find a plan with (provably) minimum length.

3. **PB'07.** Pseudo-Boolean optimization is the task of minimizing a function of zero-one variables subject to algebraic constraints, also known as zero-one integer programming. On many benchmarks, pseudo-Boolean optimizers (which are usually based on SAT solvers) outperform general integer programming packages such as CPLEX [1]. The PB'07 evaluation included both optimization and decision (feasibility) problems from a large number of domains, including formal verification and logic synthesis.

Our experiments for each solver competition followed a common procedure. First, we determined the value of $T\left(h,x\right)$ for each heuristic h and benchmark instance x using data available on the competition web site (we did not actually run any of the heuristics). The heuristics considered in these competitions are deterministic (or randomized, but run with a fixed random seed), so $T\left(h,x\right)$ is simply a single numeric value. If a heuristic did not finish within the competition time limit, then $T\left(h,x\right)$ is undefined. Second, we discarded any instances that none of the heuristics could solve within the time limit.

Given a schedule S and instance x, we will not generally be able to determine the true value of $T\left(S,x\right)$, due to the fact that $T\left(h,x\right)$ is undefined for some heuristics. We can, however, determine the value of $\min\left\{B, T\left(S,x\right)\right\}$, where B is the competition time limit. We use this lower bound in all the comparisons that follow.

4.2 Number of training instances required in practice

In this section we investigate how the number of available training instances affects the quality of a schedule computed using those training instances. To do so, we adopted the following procedure. Given a set of n instances, we select $m < n$ training instances at random, then use the greedy algorithm from §2.1 to compute an approximately optimal schedule[3] for the training instances. We then use this schedule to solve each of

[3]For all solver competitions, the number of heuristics was large enough that computing an optimal schedule via dynamic programming was impractical.

the $n-m$ remaining instances, and record the average CPU time it requires. We examined all values of m that were powers of 2 less than n. For each value of m, we repeated the experiment 100 times and averaged the results.

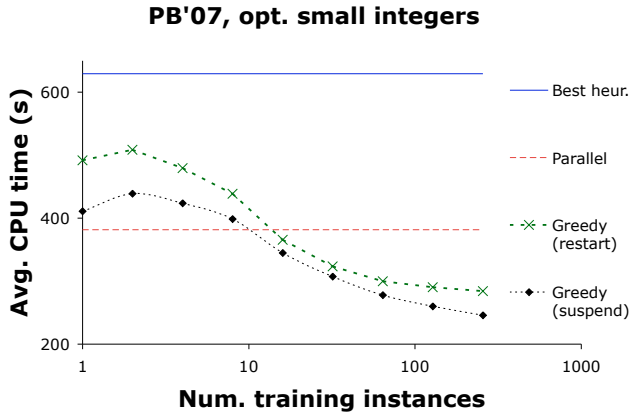


Figure 2: Experimental results for PB'07 data.

Figure 2 depicts the results for optimization problems from the "small integers" track of the PB'07 competition. The figure shows average CPU time (on test instances) as a function of the number of training instances, for both versions of the greedy algorithm (suspend-and-resume and restart). For comparison, the figure also shows the average CPU time required by the fastest individual solver, as well as a schedule that simply ran all solvers in parallel (i.e., if there are k solvers, each one receives a $\frac{1}{k}$ fraction of the CPU time).

Figure 2 has several noteworthy features. First, only a small number of training instances (in this case 16) are required in order to produce a schedule that outperforms both the fastest individual solver and the naïve parallel schedule. Second, with a sufficient number of training instances, the gap between the performance of the greedy schedules and that of the fastest individual solver is significant (in this case, more than a factor of 2). Third, the suspend-and-resume model offers only a relatively small advantage over the restart model. We have observed these same three trends in a number of other cases (e.g., see Figure 3).

We note that previous work (e.g., [16]) gave learning-theoretic bounds on the number of training instances required to learn a near-optimal schedule; however, these worst-case upper bounds are quite pessimistic relative to our experimental results.

4.3 Exploiting features

We now examine the benefit of using Boolean features to help decide which schedule to use for solving a particular problem instance. We present results for two instance sets: the *random* category of the SAT 2007 competition, and the optimal planning track of IPC-5. For the SAT instances, we labeled each instance with Boolean features based on the size of the formula, the ratio of clauses to variables, and the number of literals per clause. For the planning instances, we used features based on the planning domain, the number of goals, the number of objects, and the number of predicates in the initial conditions.

To evaluate the effect of features, we used a procedure similar to the one used in the experiments summarized in Figure 2. Given a data set, we sample m training instances at random, and examine how average performance (on test instances) varies as a function of m. For each value of m, we again repeated the experiment 100 times and averaged the results. In addition to evaluating the greedy algorithm from §2.1, we now evaluate two other approaches. The first approach, which we refer to as "Greedy w/features", uses the algorithm $\mathbf{OG}^{se}$ from §3 to select (suspend-and-resume) schedules as follows. First, we run $\mathbf{OG}^{se}$ on each of the m training instances, with exploration probability $\gamma = 1$. We then run the algorithm on each of the $n-m$ test instances, with exploration probability $\gamma = 0$ (so the algorithm receives no feedback on test instances). The second approach, which we refer to as "Features only" below, is similar except that it uses the sleeping experts algorithm of [2] to select a *single* heuristic (rather than a schedule), and runs that heuristic until it obtains a solution. Here we focus on performance as a function of the number of training instances, because the number of benchmark instances was typically too small to allow for good performance in the online setting of §2.2.

Figures 3 (A) and (B) present our results for the SAT and planning instances, respectively. Both graphs exhibit two noteworthy features. First, when the number of training instances is relatively small, a pure scheduling approach outperforms a purely feature-based approach; but as the number of training instances increases, the reverse is true. This behavior makes intuitive sense: when the number of training instances is small, committing to a single heuristic based on the training data is a very risky thing to do, and thus a purely feature-based approach can perform very poorly (e.g., worse than the naïve parallel schedule); as the number of training instances increases this becomes less of a risk. Second, in all cases, an approach that uses features to select schedules outperforms either a pure scheduling or purely feature-based approach.

Figure 4 depicts the (suspend-and-resume) schedule returned by the greedy algorithm when all available

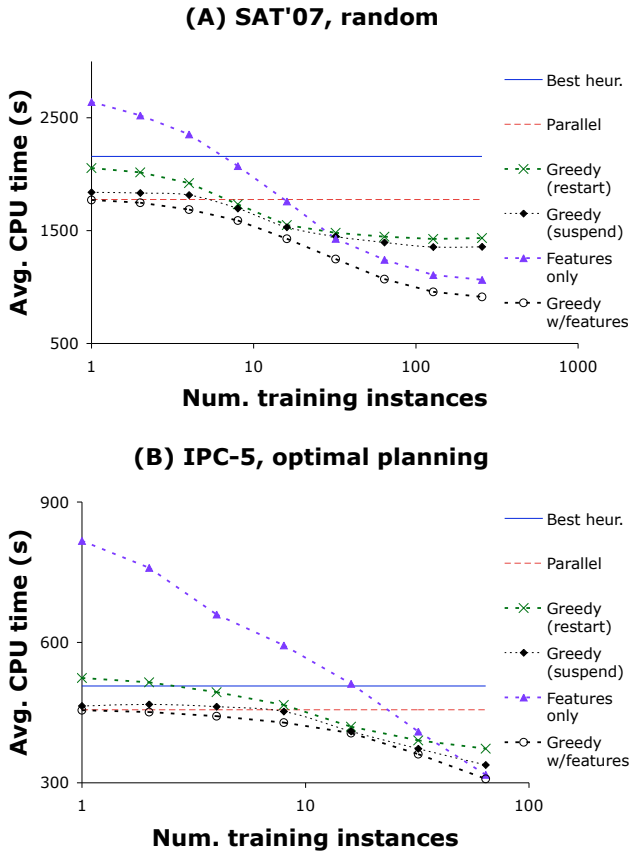


Figure 3: Experimental results for (A) SAT'07 data, *random* category and (B) IPC-5, optimal track.

SAT instances are used as training data. As indicated in the figure, the greedy schedule makes use a variety of different SAT solvers, and spends a significant amount of time running solvers whose overall average CPU time did not put them at the top of the competition.

5 Combining Anytime Algorithms

Thus far, we have thought of a heuristic as a program that, given a problem instance, runs for some fixed amount of time before definitively solving it (e.g., by returning a provably optimal solution). Now suppose instead that our heuristics are anytime algorithms that return solutions of increasing quality over time. In this case, we would like to construct a schedule that yields near-optimal solutions quickly, in addition to yielding provably optimal solutions quickly.

One simple way to do this is as follows. Define, for each instance, a set of objectives to achieve (e.g., finding a solution with cost at most α times optimal, for each $\alpha \in \{2, 1.5, 1.01\}$). For simplicity, consider the offline setting described in §2.1. For each training instance x,

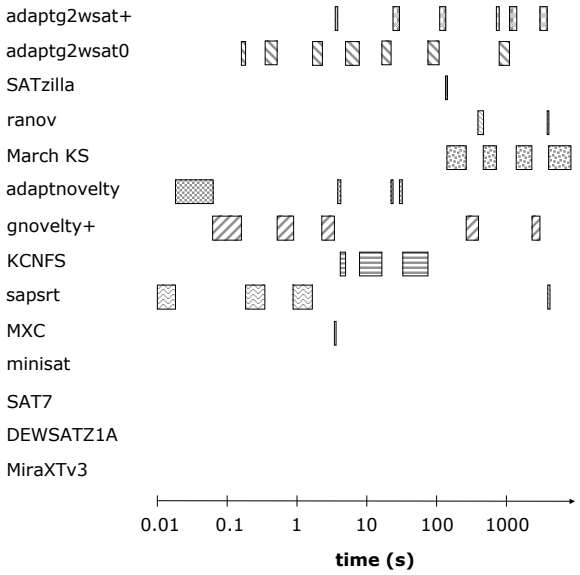


Figure 4: Schedule for SAT'07 data, *random* category. The solvers are listed in ascending order of (the lower bound on) average CPU time.

create a new set of fictitious instances $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_k$, one for each of the k objectives. For each heuristic h, define $T(h, \tilde{x}_i)$ to be the time that h requires to achieve the i^{th} objective. Then, the average time a schedule or heuristic takes to "solve" the fictitious instances is simply the average time it takes to achieve each of the k objectives on the original instances. If some objectives are more important than others, we can weight the fictitious instances accordingly (the results described in §2 readily extend to weighted sets of instances).

To evaluate this approach, we revisit the experiments performed in §4 using the PB'07 competition data, but now we measure the performance of a schedule as the average of (*i*) the time the schedule takes to find a feasible solution, (*ii*) the time the schedule takes to find an optimal solution, and (*iii*) the time the schedule takes to prove optimality (or to prove that the problem is infeasible).

Table 1 summarizes the results of these experiments. For each track of the PB'07 competition and for each of the three objectives, we define a *speedup factor* equal to the (lower bound on) average CPU time required by the fastest individual heuristic to achieve that objective, divided by the corresponding quantity for the (suspend-and-resume) greedy schedule, where the greedy algorithm is evaluated under leave-one-out cross-validation. Note that in general, the three different speedup factors listed for each track represent a comparison against three *different* heuristics.

Table 1 shows that for all three tracks, we were able to

generate a schedule that *simultaneously* outperformed each of the original heuristics in terms of each of the three objectives we considered. The results of these experiments could potentially be improved by using features[4] as in §4.3, and by sharing upper and lower bounds on the optimal objective function value among heuristics as they are discovered.

Table 1: Speedup factors for experiments with anytime algorithms, using PB'07 data.

Track	**Speedup** (prove opt)	**Speedup** (find opt)	**Speedup** (find feas)
Sm. ints	2.5	2.9	3.7
Sm. ints non-linear	1.6	1.3	1.4
Big ints.	1.2	1.5	1.4

6 Related Work

Previous work on algorithm portfolio design has almost always focused on a single aspect of the problem. In particular, almost all previous theoretical work has focused on the scheduling aspect of the problem, whereas the bulk of the experimental work has focused on the machine learning aspect of the problem. We now discuss previous work on each of these two aspects of the problem in greater detail.

6.1 Scheduling approaches

A number of papers have considered the problem of coming up with a schedule for allocating time to runs of one or more algorithms.

The earliest work on this problem measured the performance of a schedule in terms of its competitive ratio (i.e., the time required to solve a given problem instance using the schedule, divided by the time required by the optimal schedule for that instance). Results of this work include the universal restart schedule of Luby *et al.* [11] and the schedule of Kao *et al.* [9] for allocating time among multiple deterministic algorithms subject to memory constraints.

Subsequent work focused on developing schedules tailored to a particular class of problems. Gomes *et al.* [7] demonstrated that (then) state-of-the-art heuristics for Boolean satisfiability and constraint satisfaction could be dramatically improved by randomizing the heuristic's decision-making heuristics and running the randomized heuristic with an appropriate restart schedule. Huberman *et al.* [8] and Gomes *et al.* [6] combined multiple algorithms into a portfolio by running each algorithm in parallel at equal strength and assigning each algorithm a fixed restart threshold.

To fully realize the power of this approach, one must solve the problem of computing a schedule that performs well on average over a given set of problem instances collected as training data. Independently, Petrik and Zilberstein [12] and Sayag *et al.* [13] addressed this problem for two classes of schedules: task-switching schedules and resource-sharing schedules. For each of these two classes of schedules, the problem of computing an optimal schedule is NP-hard, and accordingly their algorithms have exponential running time (as a function of the number of algorithms being scheduled). Recently, Streeter *et al.* [16] presented a polynomial-time 4 approximation algorithm for computing task-switching schedules, as reviewed in §2.1.

6.2 Machine learning approaches

Another approach to algorithm portfolio design is to use features of instances to attempt to predict which algorithm will run the fastest on a given instance, and then simply run that algorithm exclusively. As an example of this approach, Leyton-Brown *et al.* [10] use least squares regression to estimate the running time of each algorithm based on quickly-computable instance features, and then run the algorithm with the smallest predicted running time. Xu *et al.* [18] presented an improved version of this approach that used a two-step prediction scheme in which the answer to a decision problem is predicted using a binary classifier, and run times are then estimated conditioned on the classifier's prediction.

6.3 Integrated approaches

In addition to the work just described, there has been previous work that addresses both the scheduling and machine learning aspects of the algorithm portfolio design problem simultaneously. For example, Gagliolo and Schmidhuber [4] presented an approach for allocating CPU time among heuristics in an online setting, based on statistical models of the behavior of the heuristics. Although their approach has no rigorous performance guarantees and would not perform well in the worst-case online setting considered in this paper, it would be interesting to compare their approach to ours experimentally.

7 Conclusions

This paper presented a new technique for addressing the scheduling and machine learning aspects of

[4]We do not present experiments that use features in conjunction with the PB'07 data because we could not readily find a suitable set of features.

the algorithm portfolio design problem, and evaluated the technique experimentally. Our main experimental findings can be summarized as follows.

1. In a number of well-studied problem domains, existing state-of-the-art heuristics can be combined into a new and faster heuristic simply by collecting a few dozen training instances and using them to compute a schedule for interleaving the execution of the existing heuristics.
2. State-of-the-art anytime algorithms for solving optimization problems can be combined, via a schedule, into an algorithm with better anytime performance.
3. Instance-specific features can be used to generate a custom schedule for a particular problem instance. Using this approach can result in better performance than using either a pure scheduling approach or a purely feature-based approach.

As suggested in §1, our experimental results could potentially be improved in at least two ways. First, we could attempt to predict a heuristic's remaining running time based on its current state and adapt our schedule accordingly. Second, we could share information among heuristics during the process of solving an instance (e.g., when solving optimization problems, the heuristics could share upper and lower bounds on the optimal objective function value).

Acknowledgements

Many thanks to Avrim Blum for suggesting the use of a sleeping experts algorithm in our problem setting. This research was supported in part by DARPA under Contract # FA8750-05-C-0033, and by the CMU Robotics Institute.

References

[1] Fadi A. Aloul, Arathi Ramani, Igor L. Markov, and Karem A. Sakallah. Generic ILP versus specialized 0-1 ILP: An update. In *ICCAD*, pages 450–457, 2002.

[2] Avrim Blum and Yishay Mansour. From external to internal regret. *Journal of Machine Learning Research*, 8:1307–1324, 2007.

[3] Uriel Feige, László Lovász, and Prasad Tetali. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004.

[4] Matteo Gagliolo and Jürgen Schmidhuber. Dynamic algorithm portfolios. In *AIMATH*, 2006.

[5] Carla P. Gomes and Bart Selman. Algorithm portfolio design: Theory vs. practice. In *UAI*, pages 190–197, 1997.

[6] Carla P. Gomes and Bart Selman. Algorithm portfolios. *Artificial Intelligence*, 126:43–62, 2001.

[7] Carla P. Gomes, Bart Selman, and Henry Kautz. Boosting combinatorial search through randomization. In *AAAI*, pages 431–437, 1998.

[8] Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg. An economics approach to hard computational problems. *Science*, 275:51–54, 1997.

[9] Ming-Yang Kao, Yuan Ma, Michael Sipser, and Yiqun Yin. Optimal constructions of hybrid algorithms. In *SODA*, pages 372–381, 1994.

[10] Kevin Leyton-Brown, Eugene Nudelman, Galen Andrew, James McFadden, and Yoav Shoham. Boosting as a metaphor for algorithm design. In *CP*, pages 899–903, 2003.

[11] Michael Luby, Alistair Sinclair, and David Zuckerman. Optimal speedup of Las Vegas algorithms. *Information Processing Letters*, 47:173–180, 1993.

[12] Marek Petrik and Shlomo Zilberstein. Learning parallel portfolios of algorithms. *Annals of Mathematics and Artificial Intelligence*, 48(1-2):85–106, 2006.

[13] Tzur Sayag, Shai Fine, and Yishay Mansour. Combining multiple heuristics. In *STACS*, pages 242–253, 2006.

[14] Matthew Streeter. *Using Online Algorithms to Solve NP-Hard Problems More Efficiently in Practice*. PhD thesis, Carnegie Mellon University, 2007.

[15] Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. Technical Report CMU-CS-07-171, Carnegie Mellon University, 2007.

[16] Matthew Streeter, Daniel Golovin, and Stephen F. Smith. Combining multiple heuristics online. In *AAAI*, pages 1197–1203, 2007.

[17] Matthew Streeter, Daniel Golovin, and Stephen F. Smith. Restart schedules for ensembles of problem instances. In *AAAI*, pages 1204–1210, 2007.

[18] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla07: The design and analysis of an algorithm portfolio for SAT. In *CP*, pages 712–727, 2007.

Dyna-Style Planning with Linear Function Approximation and Prioritized Sweeping

Richard S. Sutton, Csaba Szepesvári, Alborz Geramifard, Michael Bowling
Reinforcement Learning and Artificial Intelligence Laboratory
Department of Computing Science, University of Alberta, Edmonton, AB Canada T6G 2E8

Abstract

We consider the problem of efficiently learning optimal control policies and value functions over large state spaces in an online setting in which estimates must be available after each interaction with the world. This paper develops an explicitly model-based approach extending the Dyna architecture to linear function approximation. Dyna-style planning proceeds by generating imaginary experience from the world model and then applying model-free reinforcement learning algorithms to the imagined state transitions. Our main results are to prove that linear Dyna-style planning converges to a unique solution independent of the generating distribution, under natural conditions. In the policy evaluation setting, we prove that the limit point is the least-squares (LSTD) solution. An implication of our results is that prioritized-sweeping can be soundly extended to the linear approximation case, backing up to preceding features rather than to preceding states. We introduce two versions of prioritized sweeping with linear Dyna and briefly illustrate their performance empirically on the Mountain Car and Boyan Chain problems.

1 Online learning and planning

Efficient decision making when interacting with an incompletely known world can be thought of as an online learning and planning problem. Each interaction provides additional information that can be used to learn a better model of the world's dynamics, and because this change could result in a different action being best (given the model), the planning process should be repeated to take this into account. However, planning is inherently a complex process; on large problems it not possible to repeat it on every time step without greatly slowing down the response time of the system. Some form of incremental planning is required that, though incomplete on each step, still efficiently computes optimal actions in a timely manner.

The Dyna architecture (Sutton 1990) provides an effective and flexible approach to incremental planning while maintaining responsiveness. There are two ideas underlying the Dyna architecture. One is that planning, acting, and learning are all continual, operating as fast as they can without waiting for each other. In practice, on conventional computers, each time step is shared between planning, acting, and learning, with proportions that can be set arbitrarily according to available resources and required response times.

The second idea underlying the Dyna architecture is that learning and planning are similar in a radical sense. Planning in the Dyna architecture consists of using the model to generate imaginary experience and then processing the transitions of the imaginary experience by model-free reinforcement learning algorithms as if they had actually occurred. This can be shown, under various conditions, to produce exactly the same results as dynamic-programming methods in the limit of infinite imaginary experience.

The original papers on the Dyna architecture and most subsequent extensions (e.g., Singh 1992; Peng & Williams 1993; Moore & Atkeson 1993; Kuvayev & Sutton 1996) assumed a Markov environment with a tabular representation of states. This table-lookup representation limits the applicability of the methods to relatively small problems. Reinforcement learning has been combined with function approximation to make it applicable to vastly larger problems than could be addressed with a tabular approach. The most popular form of function approximation is linear function approximation, in which states or state-action pairs are first mapped to feature vectors, which are then mapped in a linear way, with learned parameters, to value or next-state estimates. Linear methods have been used in many of the successful large-scale applications of reinforcement learning (e.g., Silver, Sutton & Müller 2007; Schaeffer, Hlynka & Jussila 2001). Linear function approximation is also simple, easy to understand, and possesses some of the strongest convergence and performance guarantees among function approximation methods. It is

natural then to consider extending Dyna for use with linear function approximation, as we do in this paper.

There has been little previous work addressing planning with linear function approximation in an online setting. Paduraru (2007) treated this case, focusing mainly on sampling stochastic models of a cascading linear form, but also briefly discussing deterministic linear models. Degris, Sigaud and Wuillemin (2006) developed a version of Dyna based on approximations in the form of dynamic Bayes networks and decision trees. Their system, SPITI, included online learning and planning based on an incremental version of structured value iteration (Boutilier, Dearden & Goldszmidt 2000). Singh (1992) developed a version of Dyna for variable resolution but still tabular models. Others have proposed linear least-squares methods for policy evaluation that are efficient in the amount of data used (Bradtke & Barto 1996; Boyan 1999, 2002; Geramifard, Bowling & Sutton 2006). These methods can be interpreted as forming and then planning with a linear model of the world's dynamics, but so far their extensions to the control case have not been well suited to online use (Lagoudakis & Parr 2003; Peters, Vijayakumar & Schaal 2005; Bowling, Geramifard, & Wingate 2008), whereas our linear Dyna methods are naturally adapted to this case. We discuss more specifically the relationship of our work to LSTD methods in a later section. Finally, Atkeson (1993) and others have explored linear, learned models with off-line planning methods suited to low-dimensional continuous systems.

2 Notation

We use the standard framework for reinforcement learning with linear function approximation (Sutton & Barto 1998), in which experience consists of the time indexed stream $s_0, a_0, r_1, s_1, a_1, r_2, s_2, \ldots$, where $s_t \in \mathcal{S}$ is a state, $a_t \in \mathcal{A}$ is an action, and $r_t \in \mathbb{R}$ is a reward. The actions are selected by a learning agent, and the states and rewards are selected by a stationary environment. The agent does not have access to the states directly but only through a corresponding feature vector $\phi_t \in \mathbb{R}^n = \phi(s_t)$. The agent selects actions according to a policy, $\pi : \mathbb{R}^n \times \mathcal{A} \to [0, 1]$ such that $\sum_{a \in \mathcal{A}} \pi(\phi, a) = 1, \ \forall \phi$. An important step towards finding a good policy is to estimate the value function for a given policy (policy evaluation). The value function is approximated as a linear function with parameter vector $\theta \in \mathbb{R}^n$:

$$\theta^\top \phi(s) \approx V^\pi(s) = E_\pi \left\{ \sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s_0 = s \right\},$$

where $\gamma \in [0, 1)$. In this paper we consider policies that are greedy or ϵ-greedy with respect to the approximate state-value function.

Algorithm 1 : Linear Dyna for policy evaluation, with random sampling and gradient-descent model learning

Obtain initial ϕ, θ, F, b
For each time step:
  Take action a according to the policy. Receive r, ϕ'
  $\theta \leftarrow \theta + \alpha[r + \gamma\theta^\top\phi' - \theta^\top\phi]\phi$
  $F \leftarrow F + \alpha(\phi' - F\phi)\phi^\top$
  $b \leftarrow b + \alpha(r - b^\top\phi)\phi$
  $temp \leftarrow \phi'$
  Repeat p times (planning):
    Generate a sample ϕ from some distribution μ
    $\phi' \leftarrow F\phi$
    $r \leftarrow b^\top\phi$
    $\theta \leftarrow \theta + \alpha[r + \gamma\theta^\top\phi' - \theta^\top\phi]\phi$
  $\phi \leftarrow temp$

3 Theory for policy evaluation

The natural place to begin a study of Dyna-style planning is with the policy evaluation problem of estimating a state-value function from a linear model of the world. The model consists of a forward transition matrix $F \in \mathbb{R}^n \times \mathbb{R}^n$ (incorporating both environment and policy) and an expected reward vector $b \in \mathbb{R}^n$, constructed such that $F\phi$ and $b^\top\phi$ can be used as estimates of the feature vector and reward that follow ϕ. A Dyna algorithm for policy evaluation goes through a sequence of planning steps, on each of which a starting feature vector ϕ is generated according to a probability distribution μ, and then a next feature vector $\phi' = F\phi$ and next reward $r = b^\top\phi$ are generated from the model. Given this imaginary experience, a conventional model-free update is performed, for example, according to the linear TD(0) algorithm (Sutton 1988):

$$\theta \leftarrow \theta + \alpha(r + \gamma\theta^\top\phi' - \theta^\top\phi)\phi, \tag{1}$$

or according to the residual gradient algorithm (Baird 1995):

$$\theta \leftarrow \theta + \alpha(r + \gamma\theta^\top\phi' - \theta^\top\phi)(\phi - \gamma\phi'), \tag{2}$$

where $\alpha > 0$ is a step-size parameter. A complete algorithm using TD(0), including learning of the model, is given in Algorithm 1.

3.1 Convergence and fixed point

There are two salient theoretical questions about the Dyna planning iterations (1) and (2): Under what conditions on μ and F do they converge? and What do they converge to? Both of these questions turn out to have interesting answers. First, note that the convergence of (1) is in question in part because it is known that linear TD(0) may diverge if the distribution of starting states during training does not match the distribution created by the normal dynamics of

the system, that is, if TD(0) is used *off-policy*. This suggests that the sampling distribution used here, μ, might have to be strongly constrained in order for the iteration to be stable. On the other hand, the data here is from the model, and the model is not a general system: it is deterministic[1] and linear. This special case could be much better behaved. In fact, convergence of linear Dyna-style policy evaluation, with either the TD(0) or residual-gradient iterations, is not affected by μ, but only by F, as long as μ exercises all directions in the full n-dimensional vector space. Moreover, not only is the fact of convergence unaffected by μ, but so is the value converged to. In fact, we show below that convergence is to a deterministic fixed point, a value of θ such that the iterations (1) and (2) leave it unchanged not just in expected value, but for every individual ϕ that could be generated by μ. The only way this could be true is if the TD error (the first expression in parentheses in each iteration) were exactly zero, that is, if

$$\begin{aligned} 0 &= r + \gamma\theta^\top\phi' - \theta^\top\phi \\ &= b^\top\phi + \gamma\theta^\top F\phi - \theta^\top\phi \\ &= (b + \gamma F^\top\theta - \theta)^\top\phi. \end{aligned}$$

And the only way that this can be true for all ϕ is for the expression in parenthesis above to be zero:

$$\begin{aligned} 0 &= b + \gamma F^\top\theta - \theta \\ &= b + (\gamma F^\top - I)\theta, \end{aligned}$$

which immediately implies that

$$\theta = (I - \gamma F^\top)^{-1} b, \tag{3}$$

assuming that the inverse exists. Note that this expression for the fixed point does not depend on μ, as promised.

If $I - \gamma F^\top$ is nonsingular, then there might be no fixed point. This could happen for example if F were an expansion, or more generally if the limit $(\gamma F)^\infty$ were not zero. These cases correspond to world models that say the feature vectors diverge to infinity over time. Failure to converge in these cases should not be considered a problem for the Dyna iterations as planning algorithms; these are cases in which the planning *problem* is ill posed. If the feature vectors diverge, then so too may the rewards, in which case the true values given the model are infinite. No real finite Markov decision process could behave in this way.

It remains to show the conditions on F under which the iterations converge to the fixed point if one exists. We prove next that under the TD(0) iteration (1), convergence is guaranteed if the numerical radius of F is less than one,[2] and then that under the residual-gradient iteration (2), convergence is guaranteed for any F as long as the fixed point exists. That F's numerical radius be less than 1 is a stronger condition than nonsingularity of $I - \gamma F^\top$, but it is similar in that both conditions pertain to the matrix trending toward expansion when multiplied by itself.

Theorem 3.1 (Convergence of linear TD(0) Dyna for policy evaluation)**.** *Consider the TD(0) iteration with a nonnegative step-size sequence* (α_k)*:*

$$\theta_{k+1} = \theta_k + \alpha_k(b^\top\phi_k + \gamma\theta_k^\top F\phi_k - \theta_k^\top\phi_k)\phi_k, \tag{4}$$

where $\theta_0 \in \mathbb{R}^n$ *is arbitrary. Assume that (i) the step-size sequence satisfies* $\sum_{k=0}^\infty \alpha_k = \infty$*,* $\sum_{k=0}^\infty \alpha_k^2 < \infty$*, (ii)* $r(F) \leq 1$*, (iii)* (ϕ_k) *are uniformly bounded i.i.d. random variables, and that (iv)* $C = \mathbb{E}\left[\phi_k\phi_k^\top\right]$ *is non-singular. Then the parameter vector* θ_k *converges with probability one to* $(I - \gamma F^\top)^{-1}b$*.*

Proof. The idea of the proof is to view the algorithm as a stochastic gradient descent method. In particular, we apply Proposition 4.1 of (Bertsekas & Tsitsiklis 1996).

Before verifying the conditions of this result, let us rewrite (4) in terms of the matrix $G = I - \gamma F$:

$$\begin{aligned} \theta_{k+1} &= \theta_k + \alpha_k(b^\top\phi_k + \theta_k^\top(\gamma F - I)\phi_k)\phi_k \\ &= \theta_k + \alpha_k(b^\top\phi_k - \theta_k^\top G\phi_k)\phi_k \\ &= \theta_k + \alpha_k s_k. \end{aligned}$$

Here s_k is defined by the last equation.

The cited proposition requires the definition of a potential function $J(\theta)$ and will allow us to conclude that $\lim_{k\to\infty}\nabla J(\theta_k) = 0$ with probability one. Let us choose $J(\theta) = 1/2\,\mathbb{E}\left[(b^\top\phi_k + \gamma\theta^\top F\phi_k - \theta^\top\phi_k)^2\right]$. Note that by our i.i.d. assumptions on the features, $J(\theta)$ is well-defined. We need to check four conditions (because the step-size conditions are automatically satisfied): (i) The nonnegativity of the potential function; (ii) The Lipschitz continuity of $\nabla J(\theta)$; (iii) The pseudo-gradient property of the expected update direction; and (iv) The boundedness of the expected magnitude of the update, more precisely that $\mathbb{E}\left[\|s_k\|_2^2|\theta_k\right] \leq O(\|\nabla J(\theta_k)\|_2^2)$. Nonnegativity is satisfied by definition and the boundedness condition (iv) is satisfied thanks to the boundedness of the features.

Let us show now that the pseudo-gradient property (iii) is satisfied. This condition requires the demonstration of a positive constant c such that

$$c\|\nabla J(\theta_k)\|_2^2 \leq -\nabla J(\theta_k)^\top\mathbb{E}\left[s_k|\theta_k\right]. \tag{5}$$

Define $\overline{s}_k = \mathbb{E}\left[s_k|\theta_k\right] = Cb - CG^\top\theta_k$. A simple calculation gives $\nabla J(\theta_k) = -G\overline{s}_k$. Hence $\|\nabla J(\theta_k)\|_2^2 = \overline{s}_k^\top G^\top G\overline{s}_k$ and $-(\nabla J(\theta_k))^\top\overline{s}_k = \overline{s}_k^\top G\overline{s}_k$. Therefore (5) is equivalent to $c\,\overline{s}_k^\top G^\top G\overline{s}_k \leq \overline{s}_k^\top G\overline{s}_k$. In order to make this true with a sufficiently small c, it suffices to show that

[1] The model is deterministic because it generates the expectation of the next feature vector; the system itself may be stochastic.

[2] The numerical radius of a real-valued square matrix A is defined by $r(A) = \max_{\|x\|_2=1} x^T Ax$.

$s^\top G s > 0$ holds for any non-zero vector s. An elementary reasoning shows that this is equivalent to $1/2(G+G^\top)$ being positive definite, which in turn is equivalent to $r(F) \leq 1$, showing that (iii) is satisfied.

Hence, we have verified all the assumptions of the cited proposition and can therefore we conclude that $\lim_{k\to\infty} \nabla J(\theta_k) = 0$ with probability one. Plugging in the expression of $\nabla J(\theta_k)$, we get $\lim_{t\to\infty}(Cb - CG^\top\theta_k) = 0$. Because C and G are invertible (this latter follows from $r(F) \leq 1$), it follows that the limit of θ_k exists and $\lim_{k\to\infty}\theta_k = (G^\top)^{-1}b = (I-\gamma F^\top)^{-1}b$. □

Several extensions of this result are possible. First, the requirement of i.i.d. sampling can be considerably relaxed. With an essentially unchanged proof, it is possible to show that the theorem remains true if the feature vectors are generated by a Markov process given that they satisfy appropriate ergodicity conditions. Moreover, building on a result by Delyon (1996), one can show that the result continues to hold even if the sequence of features is generated in an algorithmic manner, again provided that some ergodicity conditions are met. The major assumption then is that $C = \lim_{K\to\infty} 1/K \sum_{k=1}^K \phi_k\phi_k^\top$ exists and is non-singular. Further, because there is no "noise" to reject, there is no need to decay the step-sizes towards zero (the condition $\sum_{k=0}^\infty \alpha_k^2 < +\infty$ in the proofs is used to "filter out noise"). In particular, we conjecture that sufficiently small constant step-sizes would work as well (for a result of this type see Proposition 3.4 by Bertsekas & Tsitsiklis 1996).

On the other hand the requirement on the numerical radius of F seems to be necessary for the convergence of the TD(0) iteration. By studying the ODE associated with (4), we see that it is stable if and only if CG is a positive stable matrix (i.e., iff all its eigenvalues have positive real part). From this it seems necessary to require that G is positive stable. However, to ensure that CG is positive stable the strictly stronger condition that $G + G^\top$ is positive definite must be satisfied. This latter condition is equivalent to $r(F) \leq 1$.

We turn now to consider the convergence of Dyna planning using the residual-gradient Dyna iteration (2). This update rule can be derived by taking the gradient of $J(\theta, \phi_k) = (b^\top\phi_k + \gamma\theta^\top\phi_k - \theta^\top\phi_k)^2$ w.r.t. θ. Thus, as an immediate consequence of Proposition 4.1 of (Bertsekas & Tsitsiklis 1996) we get the following result:

Theorem 3.2 (Convergence of residual-gradient Dyna for policy evaluation)**.** *Assume that θ_k is updated according to*

$$\theta_{k+1} = \theta_k + \alpha_k(b^\top\phi_k + \gamma\theta_k^\top F\phi_k - \theta_k^\top\phi_k)(\phi_k - \gamma F\phi_k),$$

where $\theta_0 \in \mathbb{R}^n$ is arbitrary. Assume that the non-negative step-size sequence (α_k) satisfies the summability condition (i) of Theorem 3.1 and that (ϕ_k) are uniformly bounded i.i.d. random variables. Then the parameter vector θ_k converges with probability one to $(I-\gamma F^\top)^{-1}b$, assuming that $(I-\gamma F^\top)$ is non-singular.

Proof. As all the conditions of Proposition 4.1 of (Bertsekas & Tsitsiklis 1996) are trivially satisfied with the choice $J(\theta) = \mathbb{E}\left[J(\theta, \phi_k)\right]$, we can conclude that θ_k converges w.p.1 to the minimizer of $J(\theta)$. In the previous theorem we have seen that the minimizer of $J(\theta)$ is indeed $\theta = (I-\gamma F^\top)^{-1}b$, finishing the proof. □

3.2 Convergence to the LSTD solution

So far we have discussed the convergence of planning given a model, but we have said nothing about the relationship of the model to data, or about the quality of the resultant solution. Suppose the model were the best linear fit to a finite dataset of observed feature-vector-to-feature-vector transitions with accompanying rewards. In this case we can show that the fixed point of the Dyna updates is the least squares temporal-difference solution. This is the solution for which the mean TD(0) update is zero and is also the solution found by the LSTD(0) algorithm (Barto & Bradtke 1996).

Theorem 3.3. *Given a training dataset of feature, reward, next-state feature triples $D = [\phi_1, r_1, \phi_1', \ldots, \phi_n, r_n, \phi_n']$, let F, b be the least-squares model built on D. Assume that $C = \sum_{k=1}^n \phi_k\phi_k^\top$ has full rank. Then the solution* (3) *is the same as the LSTD solution on this training set.*

Proof. It suffices to show that the respective solution sets of the equations

$$0 = \sum_{k=1}^n \phi_k(r_k + \gamma(\phi_k')^\top\theta - \phi_k^\top\theta), \tag{6}$$

$$0 = b + (\gamma F^\top - I)\theta \tag{7}$$

are the same. This is because the LSTD parameter vectors are obtained by solving the first equation and the TD(0) Dyna solutions are derived from the second equation.

Let $D = \sum_{k=1}^n \phi_k(\phi_k')^\top$, and $\overline{r} = \sum_{k=1}^n \phi_k r_k$. A standard calculation shows that

$$F^\top = C^{-1}D \quad \text{and} \quad b = C^{-1}\overline{r}.$$

Plugging in C, D into (6) and factoring out θ shows that any solution of (6) also satisfies

$$0 = \overline{r} + (\gamma D - C)\,\theta. \tag{8}$$

If we multiply both sides of (8) by C^{-1} from the left we get (7). Hence any solution of (6) is also a solution of (7). Because all the steps of the above derivation are reversible, we get that the reverse statement holds as well. □

Algorithm 2 : Linear Dyna with PWMA prioritized sweeping (policy evaluation)

Obtain initial ϕ, θ, F, b
For each time step:
  Take action a according to the policy. Receive r, ϕ'
  $\delta \leftarrow r + \gamma\theta^\top\phi' - \theta^\top\phi$
  $\theta \leftarrow \theta + \alpha\delta\phi$
  $F \leftarrow F + \alpha(\phi' - F\phi)\phi^\top$
  $b \leftarrow b + \alpha(r - b^\top\phi)\phi$
  For all i such that $\phi(i) \neq 0$:
    For all j such that $F^{ij} \neq 0$:
      Put j on the PQueue with priority $|F^{ij}\delta\phi(i)|$
  Repeat p times while PQueue is not empty:
    $i \leftarrow$ pop the PQueue
    $\delta \leftarrow b(i) + \gamma\theta^\top Fe_i - \theta(i)$
    $\theta(i) \leftarrow \theta(i) + \alpha\delta$
    For all j such that $F^{ij} \neq 0$:
      Put j on the queue with priority $|F^{ij}\delta|$
  $\phi \leftarrow \phi'$

Algorithm 3 : Linear Dyna with MG prioritized sweeping (policy evaluation)

Obtain initial ϕ, θ, F, b
For each time step:
  Take action a according to the policy. Receive r, ϕ'
  $\delta \leftarrow r + \gamma\theta^\top\phi' - \theta^\top\phi$
  $\theta \leftarrow \theta + \alpha\delta\phi$
  $F \leftarrow F + \alpha(\phi' - F\phi)\phi^\top$
  $b \leftarrow b + \alpha(r - b^\top\phi)\phi$
  For all i such that $\phi(i) \neq 0$:
    Put i on the PQueue with priority $|\delta\phi(i)|$
  Repeat p times while PQueue is not empty:
    $i \leftarrow$ pop the PQueue
    For all j such that $F^{ij} \neq 0$:
      $\delta \leftarrow b(j) + \gamma\theta^\top Fe_j - \theta(j)$
      $\theta(j) \leftarrow \theta(j) + \alpha\delta$
      Put j on the PQueue with priority $|\delta|$
  $\phi \leftarrow \phi'$

4 Linear prioritized sweeping

We have shown that the convergence and fixed point of policy evaluation by linear Dyna are not affected by the way the starting feature vectors are chosen. This opens the possibility of selecting them cleverly so as to speed the convergence of the planning process. One natural idea—the idea behind prioritized sweeping—is to work backwards from states that have changed in value to the states that lead into them. The lead-in states are given priority for being updated because an update there is likely to change the state's value (because they lead to a state that has changed in value). If a lead-in state is updated and its value is changed, then *its* lead-in states are in turn given priority for updating, and so on. In the table-lookup context in which this idea was developed (Moore & Atkeson 1993; Peng 1993; see also Wingate & Seppi 2005), there could be many states preceding each changed state, but only one could be updated at a time. The states waiting to be updated were kept in a queue, prioritized by the size of their likely effect on the value function. As high-priority states were popped off the queue and updated, it would sometimes give rise to highly efficient sweeps of updates across the state space; this is what gave rise to the name "prioritized sweeping".

With function approximation it is not possible to identify and work backwards from individual states, but alternatively one could work backwards feature by feature. If there has just been a large change in $\theta(i)$, the component of the parameter vector corresponding to the ith feature, then one can look backwards through the model to find the features j whose components $\theta(j)$ are likely to have changed as a result. These are the features j for which the elements F^{ij} of F are large. One can then preferentially construct starting feature vectors ϕ that have non-zero entries at these j components. In our algorithms we choose the starting vectors to be the unit basis vectors e_j, all of whose components are zero except the jth, which is 1. (Our theoretical results assure us that this cannot affect the result of convergence.) Using unit basis vectors is very efficient computationally, as the vector matrix multiplication $F\phi$ is reduced to pulling out a single column of F.

There are two tabular prioritized sweeping algorithms in the literature. The first, due simultaneously to Peng and Williams (1993) and to Moore and Atkeson (1993), which we call *PWMA prioritized sweeping*, adds the predecessors of every state encountered in real experience to the priority queue whether or not the value of the encountered state was significantly changed. The second form of prioritized sweeping, due to McMahan and Gordon (2005), and which we call *MG prioritized sweeping*, puts each encountered state on the queue, but not its predecessors. For McMahan and Gordon this resulted in a more efficient planner. A complete specification of our feature-by-feature versions of these two forms of prioritized sweeping are given above, with TD(0) updates and gradient-descent model learning, as Algorithms 2 and 3. These algorithms differ slightly from previous prioritized sweeping algorithms in that they update the value function from the real experiences and not just from model-generated experience. With function approximation, real experience is always more informative than model-generated experience, which will be distorted by the function approximator. We found this to be a significant effect in our empirical experiments (Section 6).

Algorithm 4: Linear Dyna with MG prioritized sweeping and TD(0) updates (control)

Obtain initial ϕ, θ, F, b
For each time step:
  $a \leftarrow \arg\max_a \left[b_a^\top \phi + \gamma \theta^\top F_a \phi\right]$ (or ϵ-greedy)
  Take action a, receive r, ϕ'
  $\delta \leftarrow r + \gamma \theta^\top \phi' - \theta^\top \phi$
  $\theta \leftarrow \theta + \alpha \delta \phi$
  $F_a \leftarrow F_a + \alpha(\phi' - F_a \phi)\phi^\top$
  $b_a \leftarrow b_a + \alpha(r - b_a^\top \phi)\phi$
  For all i such that $\phi(i) \neq 0$:
    Put i on the PQueue with priority $|\delta \phi(i)|$
  Repeat p times while PQueue is not empty:
    $i \leftarrow$ pop the PQueue
    For all j s.t. there exists an a s.t. $F_a^{ij} \neq 0$:
      $\delta \leftarrow \max_a \left[b_a(j) + \gamma \theta^\top F_a e_j\right] - \theta(j)$
      $\theta(j) \leftarrow \theta(j) + \alpha \delta$
      Put j on the PQueue with priority $|\delta|$
  $\phi \leftarrow \phi'$

5 Theory for Control

We now turn to the full case of control, in which separate models F_a, b_a are learned and are then available for each action a. These are constructed such that $F_a\phi$ and $b_a^\top \phi$ can be used as estimates of the feature vector and reward that follow ϕ if action a is taken. A linear Dyna algorithm for the control case goes through a sequence of planning steps on each of which a starting feature vector ϕ and an action a are chosen, and then a next feature vector $\phi' = F_a\phi$ and next reward $r = b_a\phi$ are generated from the model. Given this imaginary experience, a conventional model-free update is performed. The simplest case is to again apply (1). A complete algorithm including prioritized sweeping is given in Algorithm 4.

The theory for the control case is less clear than for policy evaluation. The main issue is the stability of the "mixture" of the forward model matrices. The corollary below is stated for an i.i.d. sequence of features, but by the remark after Theorem 3.1 it can be readily extended to the case where the policy to be evaluated is used to generate the trajectories.

Corollary 5.1 (Convergence of linear TD(0) Dyna with action models)**.** *Consider the Dyna recursion (4) with the modification that in each step, instead of $F\phi_k$, we use $F_{\pi(\phi_k)}\phi_k$, where π is a policy mapping feature vectors to actions and $\{F_a\}$ is a collection of forward-model matrices. Similarly, $b^\top \phi_k$ is replaced by $b_{\pi(\phi_k)}^\top \phi_k$. As before, assume that ϕ_k is an unspecified i.i.d. process. Let (F, b) be the least squares model of π: $F = \arg\min_G \mathbb{E}\left[\|G\phi_k - F_{\pi(\phi_k)}\phi_k\|_2^2\right]$ and $b = \arg\min_u \mathbb{E}\left[\left(u^\top \phi_k - b_{\pi(\phi_k)}^\top \phi_k\right)^2\right]$ If the numerical radius of F is bounded by one, then the conclusions of Theorem 3.1 hold: the parameter vector θ_k converges with probability one to $(I - \gamma F^\top)^{-1} b$.*

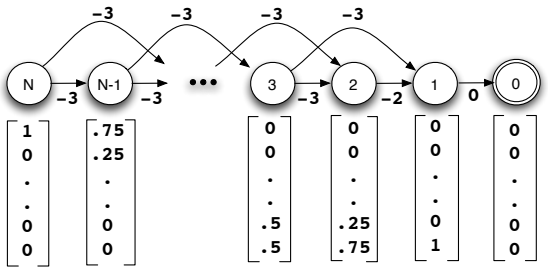


Figure 1: The general Boyan Chain problem.

Proof. The proof is immediate from the normal equation for F, which states that $\mathbb{E}\left[F\phi_k\phi_k^\top\right] = \mathbb{E}\left[F_{\pi(\phi_k)}\phi_k\phi_k^\top\right]$, and once we observe that, in the proof of Theorem 3.1, F appears only in expressions of the form $\mathbb{E}\left[F\phi_k\phi_k^\top\right]$. □

As in the case of policy evaluation, there is a corresponding corollary for the residual gradient iteration, with an immediate proof. These corollaries say that, for any policy with a corresponding model that is stable, the Dyna recursion can be used to compute its value function. Thus we can perform a form of policy iteration—continually computing an approximation to the value function for the greedy policy.

6 Empirical results

In this section we illustrate the empirical behavior of the four Dyna algorithms and make comparisons to model-free methods using variations of two standard test problems: Boyan Chain and Mountain Car. Our Boyan Chain environment is an extension of that by Boyan (1999, 2002) from 13 to 98 states, and from 4 to 25 features (Geramifard, Bowling & Sutton 2006). Figure 1 depicts this environment in the general form. Each episode starts at state $N = 98$ and terminates in state 0. For all states $s > 2$, there is an equal probability of transitioning to states $s-1$ or $s-2$ with a reward of -3. From states 2 and 1, there are deterministic transitions to states 1 and 0 with respective rewards of -2 and 0. Our Mountain Car environment is exactly as described by Sutton (1996; Sutton & Barto 1998), re-implemented in Matlab. An underpowered car must be driven to the top of a hill by rocking back and forth in a valley. The state variables are a pair (position,velocity) initialized to $(-0.5, 0.0)$ at the beginning of each episode. The reward is -1 per time step. There are three discrete actions (accelerate, reverse, and coast). We used a value function representation based on tile-coding feature vectors exactly as in Sutton's (1996) experiments, with 10 tilings over the combined (position, velocity) pair, and with the tiles hashed down to 10,000 features. In the policy evaluation experiments with this domain, the policy was to accelerate in

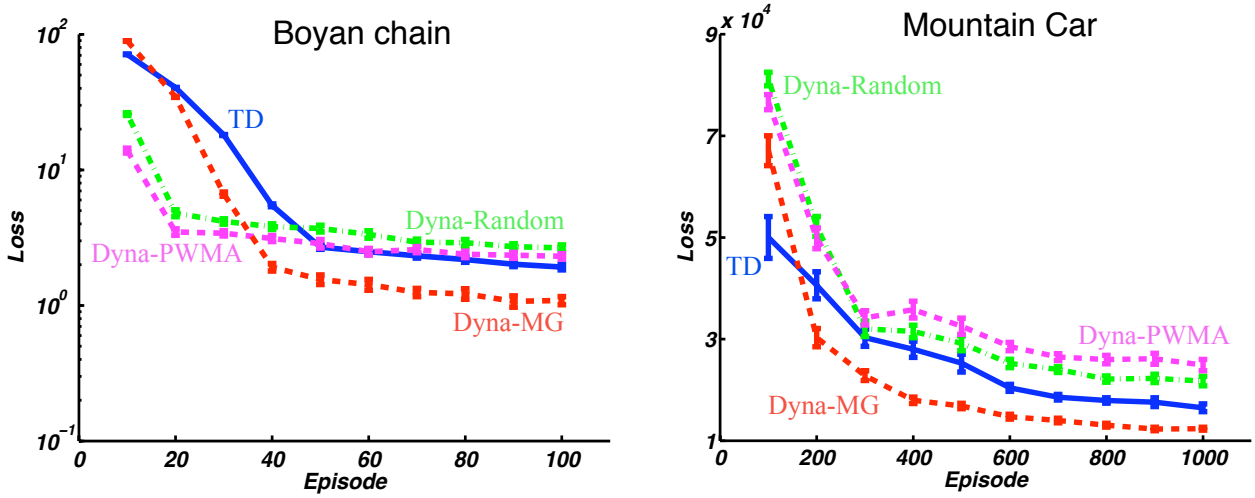


Figure 2: Performance of policy evaluation methods on the Boyan Chain and Mountain Car environments

the direction of the current velocity, and we added noise to the domain that switched the selected action to a random action with 10% probability. Complete code for our test problems as standard RL-Glue environments is available from the RL-Library hosted at the University of Alberta.

In all experiments, the step size parameter α took the form $\alpha_t = \alpha_0 \frac{N_0+1}{N_0+t^{1.1}}$, in which t is the episode number and the pair (N_0, α_0) was selected based on empirically finding the best combination out of $\alpha_0 \in \{.01, .1, 1\}$ and $N_0 \in \{100, 1000, 10^6\}$ separately for each algorithm and domain. All methods observed the same trajectories in policy evaluation. All graphs are averages of 30 runs; error bars indicate standard errors in the means. Other parameter settings were $\epsilon = 0.1$, $\gamma = 1$, and $\lambda = 0$.

We performed policy evaluation experiments with four algorithms: Dyna-Random, Dyna-PWMA, Dyna-MG (as in Algorithms 1–3), and model-free TD(0). In the case of the Dyna-Random algorithm, the starting feature vectors in planning were chosen to be unit basis vectors with the 1 in a random location. Figure 2 shows the policy evaluation performance of the four methods in the Boyan Chain and Mountain Car environments. For the Boyan Chain domain, the loss was the root-mean-squared error of the learned value function compared to the exact analytical value, averaged over all states. In the Mountain Car domain, the states are visited very non-uniformly, and a more sophisticated measure is needed. Note that all of the methods drive θ toward an asymptotic value in which the expected TD(0) update is zero; we can use the distance from this as a loss measure. Specifically, we evaluated each learned value function by freezing it and then running a fixed set of 200,000 episodes with it while running the TD(0) algorithm (but not allowing θ to actually change). The norm of the sum of the (attempted) update vectors was then computed and used as the loss. In practice, this measure can be computed very efficiently as $||A^*\theta - b^*||$ (in the notation of LSTD(0), see Bradtke & Barto 1996).

In the Boyan Chain environment, the Dyna algorithms generally learned more rapidly than model-free TD(0). Dyna-MG was initially slower than the other algorithms, then caught up and surpassed them. The relatively poor early performance of Dyna-MG was actually due to its being a better planning method. After few episodes the model tends to be of very high variance, and so therefore is the best value-function estimate given it. We tested this hypothesis by running the Dyna methods starting with a fixed, well-learned model; in this case Dyna-MG was the best of all the methods from the beginning. All of these data are for one step of planning for each real step of interaction with the world ($p = 1$). In preliminary experiments with larger values of p, up to $p = 10$, we found further improvements in learning rate of the Dyna algorithms over TD(0), and again Dyna-MG was best.

The results for Mountain Car are less clear. Dyna-MG quickly does significantly better than TD(0), but the other Dyna algorithms lag initially and never surpass TD(0). Note that, for any value of p, Dyna-MG does many more θ updates than the other two Dyna algorithms (because these updates are in an inner loop, cf. Algorithms 2 and 3). Even so, because of its other efficiencies Dyna-MG tended to run faster overall in our implementation. Obviously, there is a lot more interesting empirical work that could be done here.

We performed one Mountain Car experiment with Dyna-MG as a *control* algorithm (Algorithm 4), comparing it with model-free Sarsa (i.e., Algorithm 4 with $p = 0$). The results are shown in Figure 3. As before, Dyna-MG showed a distinct advantage over the model-free method in terms of learning rate. There was no clear advantage for either method in the second half of the experiment. We note that, asymptotically, model-free methods are never worse than model-based methods, and are often better because the model does not converge exactly to the true system because

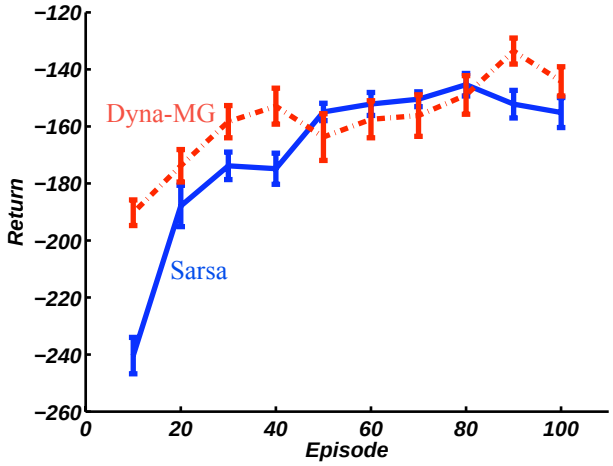


Figure 3: Control performance on Mountain Car

of structural modeling assumptions. (The case we treat here—linear models and value functions with one-step TD methods—is a rare case in which asymptotic performance of model-based and model-free methods should be identical.) The benefit of models, and of planning generally, is in rapid adaptation to new problems and situations.

These empirical results are not extensive and in some cases are preliminary, but they nevertheless illustrate some of the potential of linear Dyna methods. The results on the Boyan Chain domain show that Dyna-style planning can result in a significant improvement in learning speed over model-free methods. In addition, we can see trends that have been observed in the tabular case re-occurring here with linear function approximation. In particular, prioritized sweeping can result in more efficient learning than simply updating features at random, and the MG version of prioritized sweeping seems to be better than the PWMA version.

Finally, we would like to note that we have done extensive experimental work (not reported here) attempting to adapt least squares methods such as LSTD to online control domains, in particular to the Mountain Car problem. A major difficulty with these methods is that they place equal weight on all past data whereas, in a control setting, the policy changes and older data becomes less relevant and may even be misleading. Although we have tried a variety of forgetting strategies, it is not easy to obtain online control performance with these methods that is superior to model-free methods. One reason we consider the Dyna approach to be promising is that no special changes are required for this case; it seems to adapt much more naturally and effectively to the online control setting.

7 Conclusion

In this paper we have taken important steps toward establishing the theoretical and algorithmic foundations of Dyna-style planning with linear function approximation. We have established that Dyna-style planning with familiar reinforcement learning update rules converges under weak conditions corresponding roughly, in some cases, to the existence of a finite solution to the planning problem, and that convergence is to a unique least-squares solution independent of the distribution used to generate hypothetical experience. These results make possible our second main contribution: the introduction of algorithms that extend prioritized sweeping to linear function approximation, with correctness guarantees. Our empirical results illustrate the use of these algorithms and their potential for accelerating reinforcement learning. Overall, our results support the conclusion that Dyna-style planning may be a practical and competitive approach to achieving rapid, online control in stochastic sequential decision problems with large state spaces.

Acknowledgements

The authors gratefully acknowledge the substantial contributions of Cosmin Paduraru and Mark Ring to the early stages of this work. This research was supported by iCORE, NSERC and Alberta Ingenuity.

References

Atkeson, C. (1993). Using local trajectory optimizers to speed up global optimization in dynamic programming. *Advances in Neural Information Processing Systems, 5*, 663–670.

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 30–37.

Bertsekas, Dimitri P., Tsitsiklis. J. (1996). *Neuro-Dynamic Programming*. Athena Scientific, 1996.

Boutilier, C., Dearden, R., Goldszmidt, M. (2000). Stochastic dynamic programming with factored representations. *Artificial Intelligence 121*: 49–107.

Bowling, M., Geramifard, A., Wingate, D. (2008). Sigma point policy iteration. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems*.

Boyan, J. A. (1999). Least-squares temporal difference learning. In *Proceedings of the Sixteenth International Conference on Machine Learning*, 49–56.

Boyan, J. A. (2002). Technical update: Least-squares temporal difference learning. *Machine Learning*, 49:233–246.

Bradtke, S., Barto, A. G. (1996). Linear least-squares al-

gorithms for temporal difference learning. *Machine Learning*, 22:33-57.

Degris, T., Sigaud, O., Wuillemin, P. (2006). Learning the structure of factored markov decision processes in reinforcement learning problems. *Proceedings of the 23rd International Conference on Machine Learning.*

Delyon, B. (1996). General results on the convergence of stochastic algorithms. *IEEE Transactions on Automatic Control*, 41:1245–1255.

Geramifard, A., Bowling, M., Sutton, R. S. (2006). Incremental least-square temporal difference learning. *Proceedings of the National Conference on Artificial Intelligence*, pp. 356-361.

Kuvayev, L., Sutton, R. S. (1996). Model-based reinforcement learning with an approximate, learned model. *Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems*, pp. 101–105, Yale University, New Haven, CT.

Lagoudakis, M., Parr, R. (2003). Least squares policy iteration. *Journal of Machine Learning Research*, 4:1107-1149.

McMahan H. B., Gordon G. J. (2005). Fast exact planning in Markov decision processes. *Proceedings of the 15th International Conference on Automated Planning and Scheduling*.

Moore, A. W., Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130.

Paduraru, C. (2007). Planning with Approximate and Learned Models of Markov Decision Processes. MSc thesis, Department of Computing Science, University of Alberta.

Peng, J., Williams, R.J. (1993). Efficient learning and planning within the Dyna framework, *Adaptive Behavior 1*, 437–454.

Peters, J., Vijayakumar, S. and Schaal, S. (2005). Natural actor-critic. *Proceedings of the 16th European Conference on Machine Learning*, pp. 280-291.

Schaeffer, J., Hlynka, M., Jussila, V. (2001). Temporal difference learning applied to a high-performance game-playing program. *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 529-534.

Silver, D., Sutton, R. S., Müller, M. (2007). Reinforcement learning of local shape in the game of Go. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 1053–1058.

Singh, S. P. (1992). Reinforcement learning with a hierarchy of abstract models. *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 202–207.

Sutton, R. S. (1988). Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming, *Proceedings of the Seventh International Conference on Machine Learning*, pp. 216–224.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pp. 1038–1044. MIT Press.

Sutton, R. S., Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Wingate, D., Seppi, K. D. (2005). Prioritization methods for accelerating MDP solvers. *Journal of Machine Learning Research*, 6: 851–881.

Flexible Priors for Exemplar-based Clustering

Daniel Tarlow
Dept. of Computer Science
University of Toronto
Toronto, ON M5S 3H5

Richard S. Zemel
Dept. of Computer Science
University of Toronto
Toronto, ON M5S 3H5

Brendan J. Frey
Probabilistic and Statistical Inference Group
University of Toronto
Toronto, ON M5S 3G4

Abstract

Exemplar-based clustering methods have been shown to produce state-of-the-art results on a number of synthetic and real-world clustering problems. They are appealing because they offer computational benefits over latent-mean models and can handle arbitrary pairwise similarity measures between data points. However, when trying to recover underlying structure in clustering problems, tailored similarity measures are often not enough; we also desire control over the distribution of cluster sizes. Priors such as Dirichlet process priors allow the number of clusters to be unspecified while expressing priors over data partitions. To our knowledge, they have not been applied to exemplar-based models. We show how to incorporate priors, including Dirichlet process priors, into the recently introduced affinity propagation algorithm. We develop an efficient max-product belief propagation algorithm for our new model and demonstrate experimentally how the expanded range of clustering priors allows us to better recover true clusterings in situations where we have some information about the generating process.

1 Introduction

Clustering is a fundamental component of real-world problems in nearly every computational discipline, probably in large part due to the human tendency to use categorization as a tool for understanding data [2]. Also, clustering removes variations due to noise and replication. The value of clustering can indeed be seen by the ubiquity of the k-means algorithm and the vast amount of work on clustering that has followed [19].

Clustering is primarily used for two purposes. First, clusters provide compact approximate density representations for multimodal or difficult-to-describe distributions. Second, clustering is used to recover underlying categories in data. In many real-world problems, data points do actually come from a single unobserved class (e. g., an image pixel corresponding to an object), and we would like to group data points based on which unobserved class they come from. This second purpose motivates this work.

In order to properly describe a clustering problem, we often would like to view the data points as having come from more complex distributions than just a mixture of Gaussians in Euclidean space. For example, if we would like to cluster images while maintaining translation invariance, it is unclear how to view each image as a point in some Euclidean space [4]. In this setting, exemplar-based models are appealing, because they do not require any estimation of latent parameters, which may become difficult as spaces and distributions become more complex and high dimensional. Instead, all that is required to cluster data is a computable pairwise similarity measure between all (or a sparse subset of) pairs of points. It is often more natural to describe the clustering problem in this manner. There is an exemplar-based analog to the standard latent-mean algorithm, k-means, known as k-medians [10].

While exemplar-based models are appealing because continuous latent parameters need not be estimated, learning reduces to a combinatorial optimization problem of identifying exemplars and assigning points to exemplars. However, recent work has revealed efficient algorithms for exemplar-based clustering. Lashkari and Golland [11] give a convex formulation of an exemplar-based model that does not suffer from the initialization problems normally associated with the k-means algorithm. Affinity propagation [5] has been shown to find solutions in a matter of minutes that would take k-centers days or weeks to find and that outperform Hierarchical Agglomerative Clustering.

One drawback of existing exemplar-based methods, however, is that the implicit prior distribution over clusterings is not explicitly modeled or well-understood. In affinity propagation, for example, different granularities of clusterings are controlled by a hand-tunable parameter, called a *self-similarity* or *preference*, and there is little theoretical justification for setting this parameter.

We introduce a model that admits flexible priors into an

exemplar-based clustering framework, allowing us to express a family of flexible and infinite priors over cluster size distributions.

Since our focus is on recovering structure in data, we are interested in maximum a posteriori (MAP) inference algorithms that give a single, hard assignment as output. We develop a max-product belief propagation algorithm for our new model and show experimentally that if we have information about the generating process, without tuning any parameters in our model we are consistently able to recover true clusterings that have an unknown number of clusters and large variations in cluster sizes within individual data sets.

We further show a practical application of our model, where the priors we develop allow control over image segmentations along an axis that (to our knowledge) has been left unexplored in the image segmentation literature.

2 Background

2.1 Affinity Propagation

Affinity propagation is a clustering algorithm based on max-product belief propagation that is able to cluster data into an *a priori* unknown number of clusters.

The objective function that affinity propagation tries to maximize is:

$$P(C) = \lim_{b \to \infty} \sum_{i=1}^{N} s(i, c_i) - b \sum_{i=1}^{N} f_i(C)$$

where $C = \{c_1, \ldots, c_N\}$, $c_i \in \{1, \ldots, N\}$ is the index of the exemplar that point i is assigned to, s is a pre-computed pairwise similarity measure between all pairs of points, and $f_i(C)$ is 1 if there is some j such that $c_j = i$ and $c_i \neq i$, and 0 otherwise. Affinity propagation is an optimized max-product belief propagation algorithm over the factor graph shown in Fig. 1.

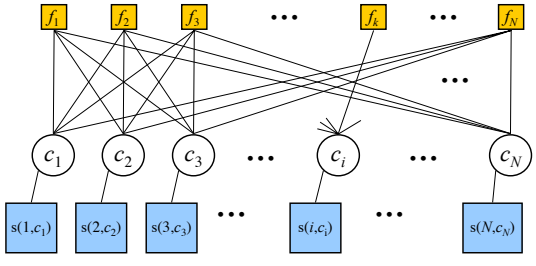


Figure 1: A factor graph representation of affinity propagation.

In affinity propagation, the prior distribution over clusterings, and thus the complexity control setting, is implicit in the user-specified *self similarity* or *preference* for each data point, represented as $s(i, i)$. The self similarity represents each point's tendency to be an exemplar. By making self similarities high, affinity propagation will find a large number of clusters, while making self similarities low will cause affinity propagation to find a small number of clusters. Though in some applications it makes sense to set these self similarities differently for different data points, there is often little reason to favor *a priori* one point to be an exemplar over another. In the latter cases, the self similarities are constrained to take on the same value for all points.

2.2 Dirichlet Process Mixture Models

Dirichlet processes provide a well-understood prior over partitions of data, which has been shown to be useful in mixture models used to tackle real-world clustering problems with an *a priori* unknown number of clusters [14]. Dirichlet process mixture models (DPMMs) use a countably infinite number of mixture components in a Bayesian framework to bypass the model selection problem of choosing the number of components [1].

We use the notation $\mathbb{1}[.]$ as the indicator function, which takes a value of 1 if the inside proposition is true and a value of 0 otherwise. For convenience, let $N_k = \sum_{i=1}^{N} \mathbb{1}[c_i = k]$ be the number of points in cluster k. Let K be the number of clusters with at least one point. After integrating out mixture weights [8], the probability of a clustering over a set of points $X = \{x_1, \ldots, x_N\}$ being given labels $C = \{c_1, \ldots, c_N\}$ is given as

$$\begin{aligned} P(C \mid X; G_0, \alpha) &= \prod_{i=1}^{N} P(x_i \mid \theta_{c_i}) \frac{\Gamma(\alpha)}{\Gamma(N+\alpha)} \alpha^K \\ &\quad \times \prod_{k=1}^{K} \Gamma(N_k) \prod_{k=1}^{K} P(\theta_k; G_0) \end{aligned}$$

where c_i is the cluster assignment for point i, α is the concentration parameter, θ_j are the latent parameters for cluster j, and G_0 is the base distribution.

In qualitative terms, Dirichlet process mixture models and affinity propagation behave similarly, in that they will continue to find more clusters in data as more data points are observed for a fixed setting of model parameters.

3 A Dirichlet Process Exemplar Model

We first develop a Dirichlet process mixture model that uses exemplars instead of latent means. We work in a *collapsed space* (i.e., where mixture weights are integrated out).

Let $X = \{X_e, X_p\}$ where X_e is the set of all points that are exemplars and X_p is the set of all points that are not exemplars. $E = \{e_1, \ldots, e_N\}$ is a set of binary variables, where $e_i = 1$ if point i is an exemplar for its cluster and 0 otherwise. The generative model is then given as follows:

- Draw a partition from a Dirichlet process prior. After

integrating out mixture weights [8], we obtain

$$P(C;\alpha) = \frac{\Gamma(\alpha)}{\Gamma(N+\alpha)}\alpha^K \prod_{k=1}^{K}\Gamma(N_k)$$

- Choose exemplars uniformly at random, but constrain there to be exactly one exemplar per group. The structure over e's is an Markov Random Field with K fully connected cliques involving all points that share the same label:

$$P(E \mid C) = \prod_{k=1}^{K}\frac{1}{Z_k} one\text{-}of\text{-}N(E_k)$$

where $one\text{-}of\text{-}N(E)$ is 1 if exactly one $e \in E$ is 1 and all the rest are zero, and zero otherwise. Let E_k be the set of all e_i such that $c_i = k$. Since there is exactly one legal choice of e's for each choice of exemplar in a group, the partition function Z_k is N_k. $P(E \mid C)$ can then be rewritten

$$P(E \mid C) = \prod_{k=1}^{K}\frac{1}{N_k}$$

as long as each non-empty group has exactly one exemplar (i.e., $\forall k \mid N_k > 0, (\sum_{i':c_{i'}=c_k}\mathbb{1}[e_{i'}=1]) = 1$), and zero otherwise.

- Draw parameters for each exemplar from G_0:

$$P(X_e;G_0) = \prod_{i=1}^{N} P(x_i;G_0)^{\mathbb{1}[e_i=1]}$$

- Draw the parameters for each remaining point from a distribution parameterized by the exemplar for its group. $P(X_p \mid X_e, C, E)$ is then given as:

$$\prod_{i=1}^{N}\prod_{j=1,j\neq i}^{N} P(x_i \mid x_j)^{\mathbb{1}[c_i=c_j \wedge e_i=0 \wedge e_j=1]}$$

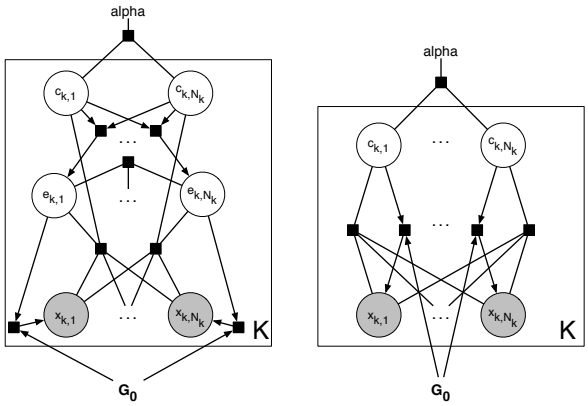
Note that in this model, x's are *not* drawn i.i.d.

The full joint likelihood, $P(C, E, X; G_0, \alpha)$ is

$$\begin{aligned} & P(C;\alpha)P(E \mid C)P(X_e;G_o)P(X_p \mid X_e, C, E) \\ = \;& \frac{\Gamma(\alpha)}{\Gamma(N+\alpha)}\alpha^K \prod_{k=1}^{K}\frac{\Gamma(N_k)}{N_k}\prod_{i=1}^{N} P(x_i;G_0)^{\mathbb{1}[e_i=1]} \\ & \times \prod_{i=1}^{N}\prod_{j=1,j\neq i}^{N} P(x_i \mid x_j)^{\mathbb{1}[c_i=c_j \wedge e_i=0 \wedge e_j=1]} \\ & \textbf{s.t. } \forall k \mid N_k > 0, \quad \sum_{i':c_{i'}=k}\mathbb{1}[e_{i'}=1] = 1 \end{aligned}$$

Since the labels C have no meaning beyond saying that points with the same label belong to the same group, we have some freedom in how to choose their values. In particular, we can constrain all groups to take on the label of their exemplar under some fixed but arbitrary ordering of points: $\forall i. c_i = i \iff e_i = 1$. Under this constraint, there is one choice of C for each legal combination of partition and set of exemplars. We can replace all references to e_i with $\mathbb{1}[c_i = i]$. The full likelihood $P(C, X; G_0, \alpha)$ then becomes

$$\begin{aligned} = \;& \frac{\Gamma(\alpha)}{\Gamma(N+\alpha)}\alpha^{\sum_{i=1}^{N}\mathbb{1}[c_i=i]} \prod_{k:c_k=k}\frac{\Gamma(N_k)}{N_k} \\ & \times \prod_{i=1}^{N} P(x_i;G_0)^{\mathbb{1}[c_i=i]} P(x_i \mid x_{c_i})^{\mathbb{1}[c_i\neq i]} \\ & \textbf{s.t. } \forall k \mid N_k > 0, \quad c_k = k \end{aligned}$$



(a) (b)

Figure 2: (a) A factor graph plate model of the Dirichlet process affinity propagation generative model. (b) After replacing constraints and reparameterizing to remove E.

Note that under this representation, the labels are not assumed to range from 1 to K. Instead, we use not necessarily consecutive labels from 1 to N since given N observed data points, there are at most N clusters in the data. However, we note that we are not explicitly truncating the model by forcing these to correspond to the first N clusters in the stick breaking representation like in [3], and there is no ordering of clusters. In this sense, we are working in the *infinite* Chinese Restaurant Process representation, rather than in a truncated approximation.

3.1 Comparison to Affinity Propagation

Affinity propagation uses the same model, but without the Dirichlet process prior over partitions, and rather than drawing exemplar parameters from a given base distribution, self similarities are set as desired by the user.

4 Dirichlet Process Affinity Propagation

In order to derive Dirichlet process affinity aropagation (DPAP), a max-product belief propagation algorithm for this model, we make one further change in representation that is useful for deriving extensions to affinity propagation [7]. Rather than representing each c_i as a multinomial variable with N states, we use N binary variables, $\{h_{i1}, \ldots, h_{iN}\}$ and a *one-of-N* constraint specifying that h_{ij} can only be 1 for one choice of j. Formally, $c_i = j \Leftrightarrow h_{ij} = 1$. The same algorithm can be derived without making this change of representation, but the derivations are simpler in this form.

By laying out these h variables in a 2-dimensional grid, we can express our model as a factor graph with one factor for each row, one factor for each column, and one factor for each h. Fig. 3 shows this factor graph representation that our algorithm operates on.

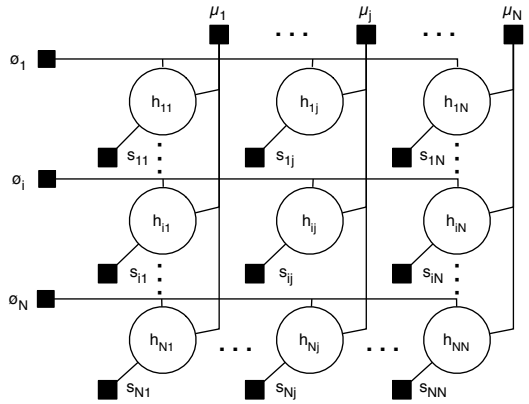


Figure 3: A factor graph representation of the Dirichlet process affinity propagation model as a grid of binary indicator variables.

$$
\begin{aligned}
\phi_i(h_{i1}, \ldots, h_{iN}) &= \mathbb{1}[\sum_{j=1}^{N} h_{ij} = 1] \\
\mu_j(h_{1j}, \ldots, h_{Nj}) &= \begin{cases} 1 & \text{if } N_j = 0 \\ \frac{\Gamma(N_j)}{N_j} \cdot \mathbb{1}[h_{jj} = 1] & \text{otherwise} \end{cases} \\
s_{ij}(h_{ij}) &= \begin{cases} P(x_i \mid x_j)^{h_{ij}} & \text{if } i \neq j \\ (\alpha \cdot P(x_j; G_0))^{h_{jj}} & \text{if } i = j \end{cases}
\end{aligned}
$$

It can be confirmed that this is equivalent to our earlier formulation.

4.1 Max-Product Belief Propagation

Max-product belief propagation is an iterative, local, message passing algorithm that can be used to find the MAP configuration of a discrete probability distribution specified by a factor graph. The algorithm was originally developed for exact inference on tree-structured graphical models, but it has empirically been shown to perform well even on graphs with cycles.

When working in log space, the algorithm is known as max-sum, and the updates for factor graphs involve either a message from a variable to each adjacent factor or a factor to each adjacent variable. The messages from variable to factor add together the messages from all adjacent factors except the factor receiving the message. Formally, if $n(x)$ is the set of all factors that share an edge with x, then the message from x to factor i, f_i, is

$$\tilde{m}_{x \to f_i}(x) = \sum_{f' \in n(x) \setminus f_i} m_{f' \to x}(x)$$

Messages from factors to variables involve a maximization over all variables in the scope of the factor except the variable receiving the message. If $X = neighbors(f)$, then

$$\tilde{m}_{f \to x}(x) = \max_{X \setminus x} \left[\log f(X) + \sum_{x' \in X \setminus x} m_{x' \to f}(x') \right]$$

We work in log space, and since all variables are binary, we normalize all messages so that $m_{X \to Y}(0) = 0$. This is equivalent to saying that all messages we pass are $m_{X \to Y}(1) = \tilde{m}_{X \to Y}(1) - \tilde{m}_{X \to Y}(0)$.

4.1.1 μ Factor Messages

The non-trivial calculation that is needed to do max-product inference in this factor graph is to compute the outgoing messages from the μ factors, $\tilde{m}_{\mu_j \to h_{ij}}(h_{ij})$. We use the notation $h_{-ij} = \{h_{i'j}\}_{i'=1, i' \neq i}^{N}$ and $h_{i-j} = \{h_{ij'}\}_{j'=1, j' \neq j}^{N}$ throughout:

$$
\begin{aligned}
&= \max_{h_{-ij}} \left[\log \mu_j(h_{:j}) + \sum_{i' : i' \neq i}^{N} m_{h_{i'j} \to \mu_j}(h_{i'j}) \right] \\
&= \max_{h_{-ij}} \begin{cases} 0 & \text{if } N_j = 0 \\ \log \frac{\Gamma(N_j)}{N_j} + \log \mathbb{1}[h_{jj} = 1] + & \\ \sum_{i' : i' \neq i}^{N} m_{h_{i'j} \to \mu_j}(h_{i'j}) & \text{otherwise} \end{cases}
\end{aligned}
$$

If $i = j$:

$$
\begin{aligned}
\tilde{m}_{\mu_j \to h_{jj}}(0) &= 0 \\
\tilde{m}_{\mu_j \to h_{jj}}(1) &= \max_{h_{-jj}} \sum_{i' : i' \neq j}^{N} m_{h_{i'j} \to \mu_j}(h_{i'j}) + \\
&\qquad \log \frac{\Gamma(1 + \sum_{i' : i' \neq j}^{N} h_{i'j})}{1 + \sum_{i' : i' \neq j}^{N} h_{i'j}}
\end{aligned}
$$

If $i \neq j$:

$$\tilde{m}_{\mu_j \to h_{ij}}(0) = \max(0, \max_{h_{-ij}} \sum_{i' : i' \neq i, i' \neq j}^{N} m_{h_{i'j} \to \mu_j}(h_{i'j}) +$$

$$m_{h_{jj}\to\mu_j}(1) + \log\frac{\Gamma(1+\sum_{i':i'\neq j,i'\neq i}^{N} h_{i'j})}{1+\sum_{i':i'\neq j,i'\neq i}^{N} h_{i'j}})$$

$$\tilde{m}_{\mu_j\to h_{ij}}(1) = \max_{h_{-ij}} \sum_{i':i'\neq i,i'\neq j}^{N} m_{h_{i'j}\to\mu_j}(h_{i'j}) + m_{h_{jj}\to\mu_j}(1) + \log\frac{\Gamma(2+\sum_{i':i'\neq j}^{N} h_{i'j})}{2+\sum_{i':i'\neq j}^{N} h_{i'j}}$$

Temporarily ignoring constants, which are irrelevant in computing the maximal settings of h's, all of the messages require a maximization of the following form:

$$\max_{h_{-ij}} \sum_{i'} m_{h_{i'j}\to\mu_j}(h_{i'j}) + \log\frac{\Gamma(\sum_{i'} h_{i'j})}{\sum_{i'} h_{i'j}}$$

This can be rewritten, using the fact that h's are binary variables, as

$$\begin{aligned} & \max_{h_{-ij}} \sum_{i'} h_{i'j}\cdot m_{h_{i'j}\to\mu_j}(1) + \\ & (1-h_{i'j})\cdot m_{h_{i'j}\to\mu_j}(0)) + \log\frac{\Gamma(\sum_{i'} h_{i'j})}{\sum_{i'} h_{i'j}} \\ = \; & \max_{h_{-ij}} \sum_{i'} h_{i'j}\cdot m_{h_{i'j}\to\mu_j}(1) + \log\frac{\Gamma(\sum_{i'} h_{i'j})}{\sum_{i'} h_{i'j}} \end{aligned}$$

We can now see that there is a tradeoff between the first term, which specifies how much a point prefers (or prefers not) to be in cluster j, and the second term, which favors more points in cluster j, regardless of how good of a fit they are. Also notice that as more points are added to cluster j, the marginal effect of the second term becomes stronger.

We can effectively remove the $\log\frac{\Gamma(K)}{K}$ term by breaking up the maximization into cases, doing the maximization for each setting of $K = 1,\ldots,N$, and then taking the largest value:

$$\begin{aligned} & \tilde{m}_{\mu_j\to h_{ij}}(h_{ij};K) + const \\ = \; & \log\frac{\Gamma(K)}{K} + \max_{h_{-ij}} \sum_{i'} h_{i'j} m_{h_{i'j}\to\mu_j}(1) \\ & \textbf{s.t.} \textstyle\sum_{i'} h_{i'j} = K \end{aligned}$$

At this point, it is easy to see that the maximum can be achieved by sorting the $m_{h_{i'j}\to\mu_j}(1)$ values in descending order, and then setting the first K h_{ij}'s to be 1 and the remainder to be 0. We do this for each value of K, and then take the setting of K that produces the largest value. The sort operation dominates the complexity, so computing a message takes $O(N\log N)$ time, which is an improvement over the $O(2^{N-1})$ time that would be needed to compute the message naively.

By sorting $m_{h_{i'j}\to\mu_j}(1)$ for all values of i, rather than for $i' : i' \neq i$, the sort operation can be shared across N instances of essentially the same computation, reducing the complexity to $O(N^2)$ for N of these maximizations that correspond to computing all outgoing messages from a single μ factor.

Intuitively, this computation is leveraging the fact that the only interaction between the h variables is via the counting term, $\log\frac{\Gamma(K)}{K}$. By conditioning on K, the terms break apart and we can maximize greedily. It should be noted that this is similar to the algorithm described in [9], though we are using it in a different way.

4.1.2 ϕ Factor Messages

The ϕ factor messages specify the $one\text{-}of\text{-}N$ constraint and can be calculated as follows:

$$\begin{aligned} \tilde{m}_{\phi_i\to h_{ij}}(h_{ij}) &= \max_{h_{i-j}} \sum_{j':j'\neq j}^{N} h_{ij'} m_{h_{ij'}\to\phi_i}(h_{ij'}) \\ & \quad \textbf{s.t.} \textstyle\sum_{j'=1}^{N} h_{ij'} = 1 \\ \tilde{m}_{\phi_i\to h_{ij}}(1) &= 0 \\ \tilde{m}_{\phi_i\to h_{ij}}(0) &= \max_{j':j'\neq j} m_{h_{ij'}\to\phi_i}(1). \end{aligned}$$

The message that we actually send is the difference:

$$m_{\phi_i\to h_{ij}}(1) = -\max_{j':j'\neq j} m_{h_{ij'}\to\phi_i}(1).$$

The rest of the messages are computed using standard max-product updates and do not require any marginalization.

4.1.3 Computing Assignments

To compute the belief for h_{ij}, we take the standard sum of incoming messages:

$$b_{h_{ij}}(h_{ij}) = s_{ij}(h_{ij}) + m_{\mu_j\to h_{ij}}(h_{ij}) + m_{\phi_i\to h_{ij}}(h_{ij}).$$

Since we enforce that $m(0) = 0$ for all messages, we set $h_{ij} = 1$ if the belief is greater than 0 and $h_{ij} = 0$ otherwise. Upon convergence, we generally do not have problems with more than one $h_{ij} = 1$ per row. However, in the case when it does happen, we set the h_{ij} with the largest belief to be 1 and all others in the row to be zero. As a final step—to refine the final assignment and to eliminate the possibility of finding an illegal solution—we run one round of iterated conditional modes (ICM) (see below), initialized with the settings for h_{ij} that we find with the technique described in this section.

4.2 Flexible Priors

At this point it is worth noting that there is nothing specific to the exact form of the Dirichlet process prior that allows us to compute μ factor messages efficiently. In particular, since the key step of the computation involves pulling the $\log\frac{\Gamma(K)}{K}$ term outside of the maximization, any unnormalized probability that is dependent only on the cluster size may be put in place here. This allows any of the priors described in [18] to be used in place of the Dirichlet process.

5 Iterated Conditional Modes

An alternative to max-product inference is to use ICM. For the purposes of this algorithm, we return to the original, expanded formulation of the model, since resampling binary variables with mutual exclusion constraints over rows would be problematic and because a sampler will mix better if it is given more flexibility in the values that labels take on. After proposing a new label for point i, we choose the best exemplars for each group, given the new labels. We then take the new label with the largest probability.

We schedule inference in a blocked manner: we iterate over each point, jointly choosing both a new value for the point currently being resampled and a new exemplar for the old and new groups simultaneously. We loop over variables to resample sequentially, until the algorithm converges.

6 Experiments

6.1 Synthetic Data

We begin by generating synthetic data from the generative process described in section 3. For these experiments, we set G_0 to be a spherical Gaussian with unit variance; $P(x_i \mid x_j) = \mathcal{N}(x_i \mid x_j, .5)$; $\alpha = 1$; and we generate 1000 data sets of 100 points each.

We compute s_{ij} for all pairs of points using the distributions given above and run three families of algorithms:

- Affinity propagation (AP(d)): We give standard AP the similarities, s_{ij}, as input, along with an additional self-similarity scaling parameter d, which is a real number that is added to each s_{ii} before running inference.
- Iterated conditional modes: We initialize the assignment to either one large group (ICM-1) or N separate groups (ICM-N), then run ICM until convergence.
- Dirichlet process affinity propagation (DPAP): The max-product inference algorithm described in section 4.

We scheduled messages in DPAP using a block synchronous schedule, where we alternated between updating mutual exclusion-based messages and cluster-based messages. We determined convergence by checking whether the largest absolute message difference between the current and previous iteration was less than 10^{-5}. DPAP converged on 94% of the runs, and produced reasonable segmentations even in the few cases when it did not converge. We used message damping of .7 for all cluster-based messages and no damping for mutual exclusion-based messages. Affinity propagation converged on all runs, and we gave it message damping of .8.

We analyze the results of running the algorithms on the synthetic data along several dimensions. First, we look at the distribution of cluster sizes found by each algorithm, aggregated over all the data sets. Fig. 4 (a) shows the distribution of cluster sizes in the true labels for the synthetic data. Fig. 4 (b) shows the distribution of cluster sizes found by affinity propagation when it is given different settings for the diagonal. This corresponds to the range of cluster size distributions that are attainable by adjusting the affinity propagation self-similarity parameter. No matter what settings are chosen for d, the algorithm is not able to simultaneously capture the heavy tail of the true labels and the large weight on small clusters of 1-5 points shown in Fig. 4 (a). Fig. 4 (c) shows that adding the infinite prior over cluster sizes allows all of the DPAP model inference algorithms to match the characteristics of the true distribution.

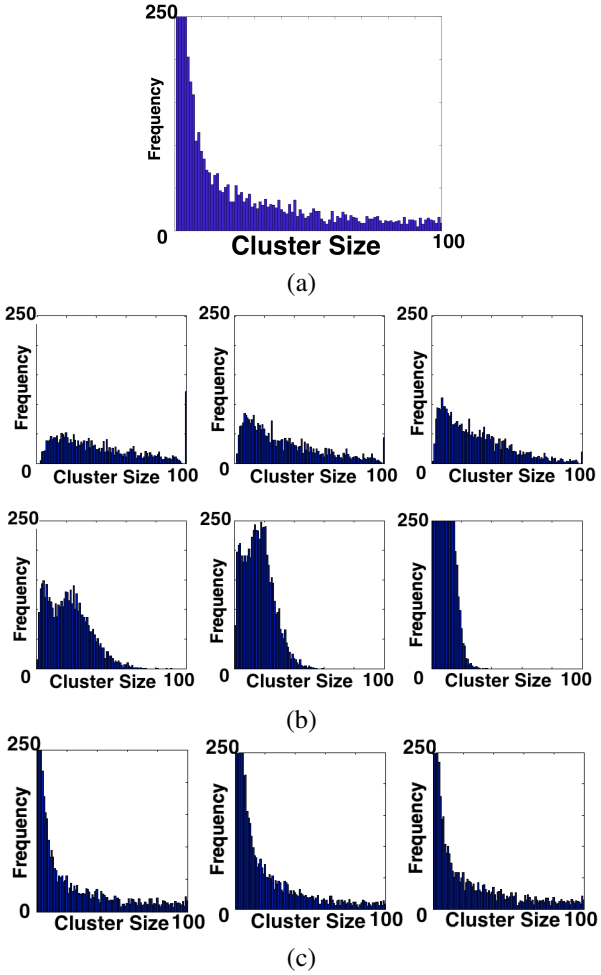
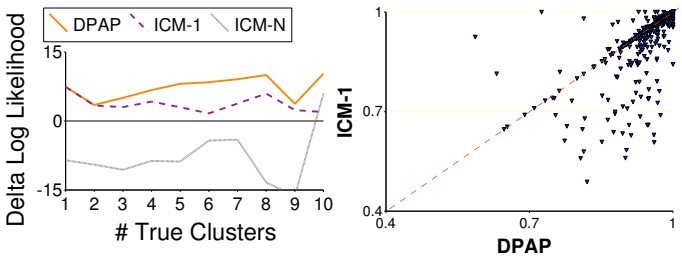


Figure 4: (a) Frequency versus true cluster size for data generated by the exemplar-based generative model, aggregated over 1000 random data sets. (b) Frequency versus size of cluster found by AP(d) with $d = -100, -50, -35; -20, -10, 0$ (c) Frequency versus size of cluster found by ICM-1, ICM-N, and DPAP, respectively.

Second, we do a comparison with respect to the log likelihoods of the different algorithms that operate on our DPAP model. Figure 5 (a) shows the average delta log like-



(a) (b)

Figure 5: (a) Comparison of log likelihoods of found labels versus number of clusters in the true labels. Scores are normalized by subtracting the likelihood of the true labels. (b) Direct comparison of the top two methods, DPAP and ICM-1, on Rand Index between the true labels and the labels found by each algorithm.

lihood of the MAP assignment found by each algorithm, as compared to the log likelihood of the true assignment, which is normalized to be the x-axis. Note that all algorithms except for ICM-N consistently find assignments with higher log likelihoods than the true labels do, which is possible due to the random nature of the generating process.

Finally, we directly compare the two best models from (a), DPAP and ICM-1, using the Rand Index [15] of the found labels against the true labels. The Rand Index is a standard metric for measuring the similarity between two clusterings, which is given as the fraction of pairs of points that are correctly assigned as being in the same cluster or correctly assigned as being in different clusters. A score of 1 means that the labels are identical up to a permutation of label names. Figure 5 (b) shows that both do quite well in general, meaning that we are able to recover the underlying labels in many cases, but DPAP consistently outperforms ICM-1, often by a substantial margin.

6.2 Real Data: Image Segmentation

Image segmentation is a clustering problem that is made easier if non-Gaussian similarity measures are used. In fact, many approaches to image segmentation do not use latent-mean based models, due in part to the restrictions that they place on the likelihood functions that may be used.

We begin by converting the image into a superpixel representation [16, 6], which has been shown to reduce the size of the problem without losing much information. We find approximately 250 superpixels for each image. We then follow in the spirit of many image segmentation algorithms and use a combination of color and boundary information to compute pairwise similarities between superpixels.

The color component of our similarity measure is based on the difference between mean colors of superpixels in RGB space:

$$s_R(i,j) = -\tau_R ||\bar{R}(i) - \bar{R}(j)||^2$$

To incorporate boundary information, we first use the local boundary detector described in [12] to find a soft edge map, E, where each pixel is assigned a probability that it is an edge pixel. For each pair of neighboring superpixels, we look at the pixels on the boundary, b, between superpixel i and superpixel j, and then take the average edge response over boundary pixels as the edge distance between the superpixel pair.

$$d_E(i,j) = \tau_E \begin{cases} \frac{\sum_{x,y \in b(i,j)} E(x,y)}{|boundary(i,j)|} & \text{if } |b(i,j)| > 0 \\ \infty & \text{otherwise} \end{cases}$$

To convert edge distances to a similarity measure, we find the shortest paths between all pairs of superpixels relative to d_E using Dijkstra's algorithm as is standard. Negating these shortest-path distances gives us the boundary component of our similarity measure, $s_E(i,j)$.

Finally, our full similarity measure simply adds together the two components:

$$s_{ij}(1) = s_R(i,j) + s_E(i,j)$$

It should be noted that we make no claim that our similarity measure between superpixels is optimal. In fact, we make the opposite claim. This was a relatively simple similarity measure that took little hand-tuning to produce. Instead, our aim is to explore whether imposing a prior over cluster sizes is an axis of control that can produce a more powerful segmentation algorithm, a question rarely considered in the image segmentation literature.

Since the focus of this work is on the prior and not image segmentation, we set the values of τ_R and τ_E by hand, making sure only that both sources of information contribute roughly equally to the full similarity measure. We leave the values constant for all experiments presented. We give the same similarity matrix to each of the following algorithms, then show results on a range of reasonable parameter settings for each of the respective algorithms:

- Dirichlet process affinity propagation (DPAP): We use DPAP, as suggested by the synthetic experiments. We set the self similarities to be constant at a level that produces an oversegmentation, then we multiply all similarities (self and non-self) by a scaling parameter. This has the effect of varying the relative strength of the prior versus image information. The concentration parameter, α, is set to 1 for all experiments.

- Affinity propagation (AP): We give standard AP the similarities, then vary the self similarity to produce a range of clusterings.

- Normalized cuts [17] (Ncut): We give normalized cuts the same similarity matrix as we give the AP and DPAP, then we vary K, the number of clusters that we ask it to find.

Figure 6 (a) shows a range of results found by each algorithm on a family scene. The image is quite crowded and has a range of different true cluster sizes. As can be seen by looking at (a6) (where a1-a12 index the color images in raster scan order), the Dirichlet process prior is beneficial, because it allows large clusters like the Christmas tree to remain in a single cluster while simultaneously finely discriminating in the smaller clusters like the faces and chair in the center of the scene . In order for either Ncut or AP to get this level of fineness, they must oversegment the Christmas tree. Compare (a6) to (a8), (a9), (a11), and (a12).

Figure 6 (b) shows segmentations found by each algorithm on a scuba diving scene. The oversegmentation of the background by both AP and Ncut is made more clear in this case, which is likely due to the implicit prior that favors clusters of roughly the same size. However, we can also see that DPAP is too willing to find small, singleton clusters. This comes from the fact that the Dirichlet process prior becomes more discriminating as the clusters become smaller. Since there are few strongly coherent regions other than the background, DPAP is willing to call nearly all of the non-water regions singletons.

Table 1 shows that the DPAP prior provides a quantitative improvement in the clusterings that we are able to find relative to the Rand Index for the two images shown, but we note that of course there are some images where such a prior is appropriate and some where it is not. The images that we have chosen to show are meant to be representative of the range of solutions that each algorithm is able to produce, rather than focusing only on illustrating the best results.

A further point is that all algorithms are too willing to form small, 1-3 superpixel clusters. This suggests that while the Dirichlet process prior is often more appropriate than the implicit priors of AP and Ncut, an ideal image segmentation prior would shift some of the weight that the Dirichlet process prior places on small clusters to medium-sized clusters for these images. Such a prior could be expressed within our model, as mentioned in Section 4.2, but we leave learning or choosing a prior or set of priors specific to image segmentation as future work.

Table 1: Quantitative Results For Real Data

Family Christmas			Scuba Diver		
Figure(Alg.)	K	R.I.	Figure(Alg.)	K	R.I.
a3(True)	21	1	b3(True)	10	1
a4(DPAP)	19	.746	b4(DPAP)	6	.734
a5(DPAP)	30	.884	**b5(DPAP)**	**15**	**.773**
a6(DPAP)	**50**	**.937**	b6(DPAP)	31	.697
a7(AP)	7	.833	b7(AP)	5	.494
a8(AP)	19	.915	b8(AP)	16	.457
a9(AP)	42	.919	b9(AP)	31	.377
a10(Ncut)	7	.862	b10(Ncut)	6	.424
a11(Ncut)	19	.914	b11(Ncut)	15	.395
a12(Ncut)	41	.917	b12(Ncut)	30	.379

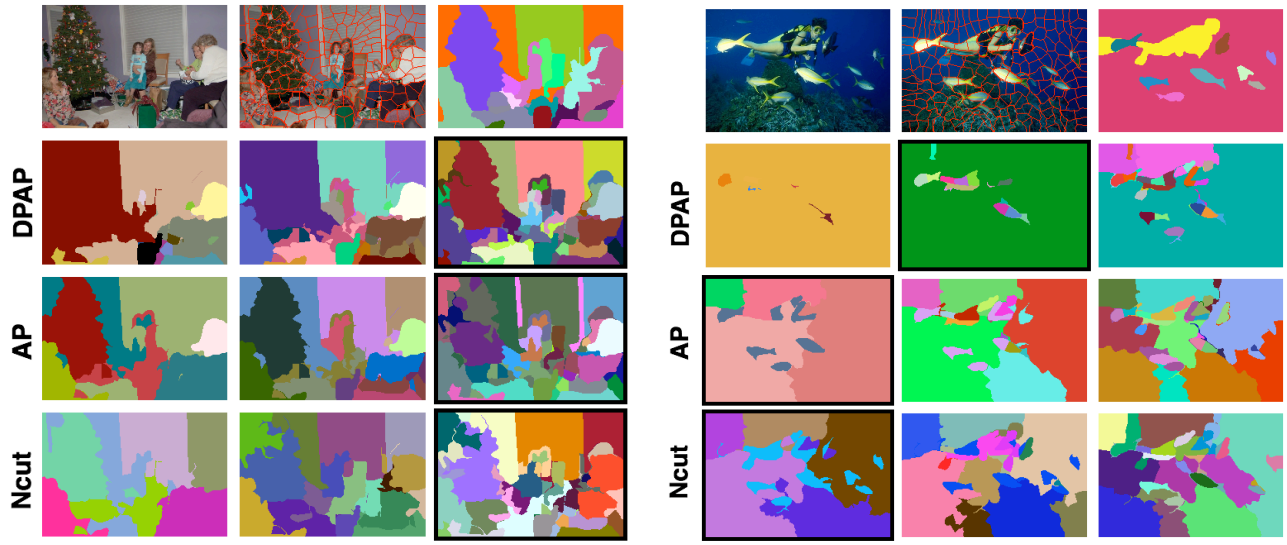
7 Discussion

We have developed a family of priors for exemplar-based clustering that are more flexible in their ability to express prior knowledge about cluster size distributions. If there is reason to believe that cluster sizes follow a certain distribution, such as in word counts, various power-law phenomena, or the size of objects in images, then this gives us the ability to impart that information into an exemplar-based clustering setting. The combination of flexible priors with exemplar-based clustering allows us to naturally represent a broad range of clustering problems. With added flexibility comes added ability to discover underlying structure. The tradeoff that we make to add flexibility in the prior information comes at an order N cost in complexity over affinity propagation, due to the fact that we must condition on all values of K inside the μ factor message computations. However, many of these computations result in the conclusion that no points should be assigned to the given cluster, so there are likely many directions by which to improve efficiency, including calculating bounds and finding intelligent message schedules.

If we have no prior information about cluster sizes, it may be reasonable to default to using methods without explicitly defined priors. On the other hand, those methods have implicit priors, which may be equally unjustified. In general, when trying to recover underlying classes of a known type, it is important to consider what assumptions about cluster sizes should be included in the model.

The results presented here show that the combination of flexible priors with exemplar-based clustering gives a large degree of control over the characteristics of the clusterings that our algorithms produce. Our work suggests other research directions along similar lines. One direction that would be interesting is to incorporate a Dirichlet process prior into the k-means component of spectral clustering.

We have illustrated our model using Dirichlet process priors on an image segmentation problem, but we emphasize that our model is broadly applicable to many different priors and applications. Further, though there has been some work revealing that max-product belief propagation may be done efficiently with cardinality-based clique potentials [9] and priors placed on node degrees [13], this is the first work that we are aware of that shows how to use Dirichlet process priors in a max-product belief propagation setting. We are currently exploring other problems where using Dirichlet process priors with max-product inference may be useful.



(a) (b)

Figure 6: Top row: Original image; Superpixellation; True Labels. Second-Fourth Row: Image segmentations of a crowded family scene and a scuba diving scene using a range of parameter settings for DPAP, AP, and Ncut, respectively. The clustering with the best Rand Index is highlighted in black. Best viewed in color. (a) As AP and Ncut move towards finer segmentations (right column), they oversegment the the Christmas tree. (b) Similarly, as AP and Ncut move towards finer segmentations, they oversegment the water. Since DPAP has an explicit prior over cluster sizes that encourages large clusters to get larger, it chooses to split apart the smaller clusters—the diver and fish—instead of the larger water cluster.

References

[1] C. Antoniak. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics*, 2:1152–1174, 1974.

[2] Aristotle. Categories. *The University of Adelaide Library*, 2007.

[3] David M. Blei and Michael I. Jordan. Variational methods for the dirichlet process. In *ICML*, 2004.

[4] D. Dueck and B.J. Frey. Non-metric affinity propagation for unsupervised image categorization. In *International Conference on Computer Vision (ICCV)*, 2007.

[5] B.J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, February 2007.

[6] A. Efros G. Mori, X. Ren and J. Malik. Recovering human body configurations: Combining segmentation and recognition. In *IEEE Computer Vision and Pattern Recognition*, 2004.

[7] I. Givoni and B. Frey. A binary model for affinity propagation. *Submitted to Neural Computation*, 2008.

[8] T. L. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process. In *Gatsby Computational Neuroscience Unit Technical Report GCNU TR 2005-001*, 2005.

[9] R. Gupta, A Diwan, and S. Sarawagi. Efficient inference with cardinality-based clique potentials. In *ICML*, volume 227, pages 329–336, 2007.

[10] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1981.

[11] D. Lashkari and P. Golland. Convex clustering with exemplar-based models. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[12] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using brightness and texture. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[13] Q. D. Morris, B. J. Frey, and C. J. Paige. Denoising and untangling graphs using degree priors. In *NIPS*, 2003.

[14] Radford Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2), 2000.

[15] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971.

[16] X. Ren and J. Malik. Learning a classification model for segmentation. In *ICCV*, 2003.

[17] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[18] Max Welling. Flexible priors for infinite mixture models. In *ICML Workshop on Nonparametric Bayesian Methods*, 2006.

[19] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16, 2005.

Propagation using Chain Event Graphs

Peter A. Thwaites
Statistics Dept.
University of Warwick
Coventry UK CV4 7AL

Jim Q. Smith
Statistics Dept
University of Warwick
Coventry UK CV4 7AL

Robert G. Cowell
Cass Business School
City University
London EC1Y 8TZ

Abstract

A Chain Event Graph (CEG) is a graphical model which is designed to embody conditional independencies in problems whose state spaces are highly asymmetric and do not admit a natural product structure. In this paper we present a probability propagation algorithm which uses the topology of the CEG to build a *transporter* CEG. Intriguingly, the transporter CEG is directly analogous to the triangulated Bayesian Network (BN) in the more conventional junction tree propagation algorithms used with BNs. The propagation method uses factorization formulae also analogous to (but different from) the ones using potentials on cliques and separators of the BN. It appears that the methods will be typically more efficient than the BN algorithms when applied to contexts where there is significant asymmetry present.

1 INTRODUCTION

Based on an event tree, a Chain Event Graph (CEG) is a more expressive alternative to a discrete Bayesian Network (BN), embodying collections of conditional independence statements in its topology. In Anderson and Smith (2008) it is shown not only how asymmetries in a problem's sample space can be represented explicitly through the topology of its CEG, but also how it can express a much wider range of types of conditional independence statement not simultaneously expressible through a single BN. As with the BN, the CEG of an hypothesised model can be interrogated using natural language before the graph is embellished with probabilities. In Thwaites and Smith (2006) and Riccomagno and Smith (2005) we demonstrate how the CEG can also be used to represent and analyse various causal hypotheses. In this paper we continue the development of CEGs by demonstrating how the graph provides a useful structure for fast probability propagation in asymmetric models.

It has been noted that the CEG is an especially powerful framework for inference when a probability model is highly asymmetric and elicited through a description of how situations unfold. Although theoretically a BN can be used in this context, the clique probability tables are then very sparse and contain many zeros or repeated probabilities. This impedes fast propagation algorithms and has led to the development of many context specific variants of BNs (Boutilier et al 1996, McAllester et al 2004, Poole and Zhang 2003, Salmeron et al 2000), often based on trees within cliques. These developments provoke the question as to whether a single tree might be used for propagation instead of the BN. Now obviously the event tree itself expresses no conditional independencies in its topology and these independencies are the building blocks of current propagation algorithms. However, unlike the event tree, the CEG expresses a fairly comprehensive collection of conditional independencies. In this paper we demonstrate the surprising fact that there is a direct analogue between a distribution on a BN expressed as a product of potentials supported by a graph of cliques and separators, and propagation algorithms on CEGs using the distributions on the children of the CEG's non-leaf nodes and marginal likelihoods on the vertices themselves. This enables us to develop fast propagation algorithms that use a single graph, the *transporter* CEG – analogous to a triangulated BN – as its framework. This framework is highly efficient for asymmetric/non-product-space contexts, and in particular does not involve propagating zeros in sparse but large probability tables, nor continually repeating the same calculations, which would be the case if we were to use the BN as a framework in this sort of environment with a naive BN propagation algorithm.

In the next section we formally define the transporter CEG $C(T)$ of a hypothesised probability tree T. In

section 3 we present an algorithm analogous to that of Cowell and Dawid (1992) for a BN where conditional probability tables associated with the children of a given vertex of the CEG take the role of cliques, and vertex probabilies take the role of separators. In section 4 we demonstrate the efficiency of this algorithm with a simple example.

2 PROBABILITY TREES AND CHAIN EVENT GRAPHS

Probability trees (Shafer 1996), and their control analogues decision trees, have been found to be a very natural and expressive framework for probability and decision problems, and they provide an excellent framework for describing sample space asymmetry and inhomogeneity in a given context (see for example French and Insua (2000)). We start with an event tree T with vertex set $V(T)$ and (directed) edge set $E(T)$. Henceforth call the tree's non-leaf vertices $\{v\}$ *situations*, and denote this set of vertices $S(T) \subset V(T)$. We can convert an event tree into a probability tree by specifying a transition matrix from its vertices $V(T)$, where the absorbing states correspond to the leaf vertices. Transition probabilities from a situation are zero except for transitions to one of that situation's children. This makes the transition matrix upper triangular. Such a matrix would look like the one in Table 1 which shows part of the matrix for the problem described in Example 1. Note that each transition probability can be identified by an edge on the tree.

Table 1: Part of the transition matrix for Example 1

	v_0	v_1	v_2	v_3	v_4^1	v_4^2	v_4^3	v_5^1	v_5^2	$\cdots$	v_∞^1	$\cdots$
v_0	0	θ_1	θ_2	θ_3	0	0	0	0	0	$\cdots$	0	$\cdots$
v_1	0	0	0	0	θ_5	0	0	0	0	$\cdots$	θ_4	$\cdots$
v_2	0	0	0	0	0	θ_6	0	θ_7	0	$\cdots$	0	$\cdots$
v_3	0	0	0	0	0	0	θ_8	0	θ_9	$\cdots$	0	$\cdots$
$\vdots$									$\vdots$		$\vdots$	

One way of seeing conditional independence statements on a BN is as identities in certain vectors of conditional probabilties – explicitly those probability vectors associated with different ancestor configurations but the same parent configuration of a variable in the BN (Riccomagno and Smith 2007). There is a large class of models where the probabilities in some of the rows of the transition matrix can be identitifed with each other. The CEG is a topological representation of this class of models, and the *transporter* CEG defined below is a subgraph of the CEG.

Let $T(v_i)$, $i = 1, 2$ be the unique subtrees whose roots are the situations v_i, and which contain all vertices after v_i in T. Say v_1 and v_2 are in the same *position* w if:

1. the trees $T(v_1)$ and $T(v_2)$ are topologically identical.
2. there is a map between $T(v_1)$ and $T(v_2)$ such that the edges in $T(v_2)$ are annotated, under that map, by the same (possibly unknown) probabilities as the corresponding edges in $T(v_1)$.

It is easily checked that the set $W(T)$ of positions w partitions $S(T)$. Furthermore, somewhat more subtlely, if $v_1, v_2 \in w$ and $v_{ij} \in V(T(v_i))$, then the vertex sets of $T(v_i)$ $i = 1, 2$ are mapped on to each other by this map, and $v_{ij} \in w_j$ $i = 1, 2$ for some position w_j (providing v_{ij} is not a leaf-vertex in either subtree). For details of this property see Anderson and Smith (2008).

We now draw a new graph to depict both the sample space of T and certain conditional independence statements. The *transporter* CEG $C(T)$ is a directed graph whose vertices $W(C(T))$ are $W(T) \cup \{w_\infty\}$. There is an edge $(e \in E(C(T)))$ from w_1 to $w_2 \neq w_\infty$ for each situation $v_2 \in w_2$ which is a child of a fixed representative $v_1 \in w_1$ for some $v_1 \in S(T)$, and an edge from w_1 to w_∞ for each leaf node $v \in V(T)$ which is a child of some fixed representative $v_1 \in w_1$ for some $v_1 \in S(T)$. The transporter CEG (henceforth labelled simply as C) is the subgraph of a CEG (defined in Anderson and Smith (2008)) where all undirected edges in the CEG are omitted. The relationship between the transporter CEG and the CEG is directly analogous to the relationship between a triangulated BN and the original BN. Certain conditional independence statements that can be lost through conditioning are simply forgotten so that an homogeneous propagation algorithm can be constructed on the basis of the enduring conditional independencies. Unlike the BN, this CEG can have many edges between two vertices and always has a single sink vertex w_∞. Although typically having many fewer vertices than T, it retains a depiction of the sample space structure of T. Thus it is easy to check that the set of root to leaf paths of the tree (representing the set of all possible unfoldings of the history of a unit) are in one to one correspondence with the set of root to sink paths on the transporter CEG. The CEG-construction process is illustrated in Example 1.

Example 1

Consider the tree in Figure 1, which has 16 atoms (root-to-leaf paths). Note that as the subtrees rooted in the vertices $\{v_4^i\}$ are the same, and those rooted in $\{v_5^i\}$ are the same, the distribution on the tree can be stored using 7 conditional tables which contain 16 (9 free) probabilities.

Our transporter CEG (Figure 2) is produced by combining the vertices $\{v_4^i\}$ into one *position* w_4, the ver-

tices $\{v_5^i\}$ into one position w_5, the vertices $\{v_6^i\}$ into one position w_6, and all leaf-vertices into a single sink-node w_∞. The full CEG for our example is *simple* – it has no undirected edges, and is identical to the transporter CEG C. For a *simple* CEG, **all** the conditional independencies inherent in the problem are conveyed by the transporter CEG.

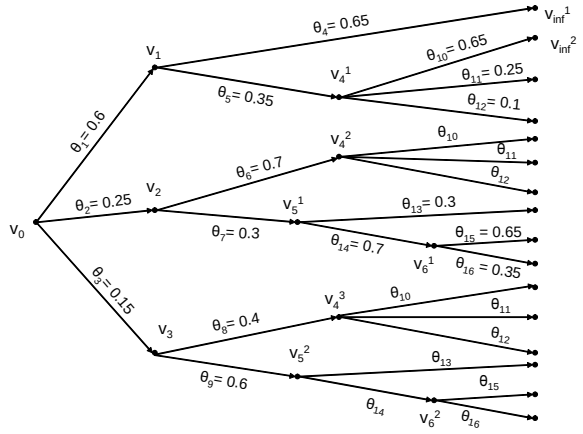


Figure 1: Tree for Example 1

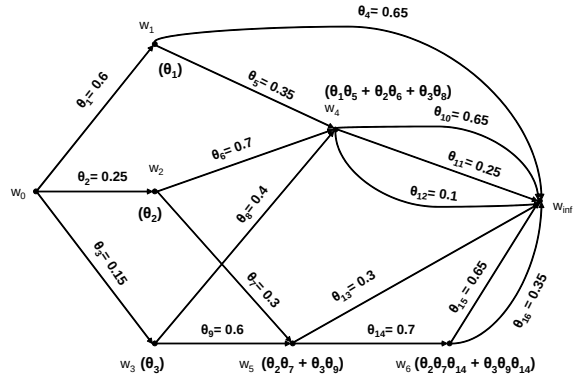


Figure 2: Transporter CEG for Example 1

Figure 2 shows the probabilities of reaching each position w (the event *reaching* w, denoted $\Lambda(w)$, is the union of all root-to-sink paths passing through w). It also shows each edge-probability $\pi_e(w' \mid w)$ $(= \pi(\Lambda(e(w,w')) \mid \Lambda(w))$, where $\Lambda(e(w,w'))$ is the union of all root-to-sink paths utilising the edge $e(w,w')$).

The problem represented by the tree in Figure 1 is asymmetric in that not all the root-to-leaf paths are of the same length, and also in the local structure associated with its vertices. We do not know whether the vertices $\{v_4^i\}$ are related in any contextual way to the vertices $\{v_5^i\}$ or $\{v_6^i\}$, and hence we cannot obviously define variables on the sigma-algebra of the tree to allow us to represent the problem as a BN. Even supposing we were able to represent the problem in such a way, the conditional independencies embodied in the problem (and in our transporter CEG) cannot be efficiently coded in a BN without introducing tables with many zeros. Consequently, even in this very simple example we have efficiency gains in storing this distribution over using a saturated model, a BN, or a tree.

3 A SIMPLE PROPAGATION ALGORITHM

3.1 THE FRAMEWORK

To specify the joint distribution of all random variables measurable with respect to a CEG we simply need to specify the vector of conditional probability mass functions associated with each of its positions. The first step of our propagation algorithm is analogous to the triangulation step for a BN, which allows us to retain all conditional independence properties at the cost of a possible loss of efficiency. To do this we ignore conditional independence statements coded by the undirected edges of the CEG and work only with the subgraph consisting of its positions, together with its directed edges, but not its undirected edges – our transporter CEG C.

For each position $w \in W = W(C) \backslash \{w_\infty\}$ we store a vector of probabilites $\boldsymbol{\pi}(w) = \{\pi_e(w' \mid w) \mid e(w,w') \in E(w)\}$ where $E(w) \subset E(C)$ is the set of all edges emanating from w. $\boldsymbol{\pi}(w)$ is of course a conditional probability distribution. We let $X(w)$ be the random variable taking values on $\{1, 2, \ldots, n(E(w))\}$ (where $n(E(w))$ is the number of edges emanating from w) whose probability mass function is given by the components of $\boldsymbol{\pi}(w)$ taken in order. The positions $w \in W$ take the role of the cliques in a triangulated BN, whilst the vectors $\{\boldsymbol{\pi}(w) \mid w \in W\}$ are analogous to the clique probability tables.

We can now specify the probability π_λ of every atom λ (a root to sink path of C, of length $n(\lambda)$) as a function of $\{\boldsymbol{\pi}(w) \mid w \in W\}$ and C. If:

$$\lambda = (w_0 = w_\lambda[0], e_\lambda[1], w_\lambda[1], \ldots, e_\lambda[n(\lambda)], w_\infty)$$

then

$$\pi_\lambda = \prod_{i=1}^{n(\lambda)} \pi(e_\lambda[i])$$

where $\pi(e_\lambda[i])$ is a component of the probability vector $\boldsymbol{\pi}(w_\lambda[i-1])$, $1 \leq i \leq n(\lambda)$. It follows that the distribution of any random variable measurable with respect to C can be calculated from $\{\boldsymbol{\pi}(w) \mid w \in W\}$.

3.2 COMPATIBLE OBSERVATIONS

Recall that propagation algorithms for BNs based on triangulation are only designed to propagate information that can be expressed in the form

$O(\mathbf{A}) = \{X_j \in A_j\}$ for some subsets $\{A_j\}$ of the sample spaces of $\{X_j\}$ the vertex-variables of the BN. Propagating information about the value of some general function of the vertex variables using local message passing is not generally possible, because conditioning on the values of such a function can destroy the conditional independencies on which the local steps of the propagation algorithm depend for their validity.

In the same way the types of observation we can efficiently propagate using C and $\{\boldsymbol{\pi}(w) \mid w \in W\}$ needs to be constrained. In general an observation can be identified with a subset Λ of the set of all root to sink paths $\{\lambda\}$. The most obvious constraining assumption on Λ (and the one we will henceforth make in this paper) about what we might learn is that our observation Λ can be identified with having learned that $\{X(w) \in A(w)\}$ for some subsets $\{A(w)\}$ of the sample spaces of the position random variables $\{X(w)\}$. Call such a set $C-$*compatible*. Note that Λ is $C-$compatible if and only if there exists possibly empty subsets $\{E_\Lambda(w) \mid w \in W\}$ such that

$$\Lambda = \{\lambda \mid e_\lambda \in E_\Lambda(w) \text{ for some } w \in W, \text{ for each edge } e_\lambda \text{ on the path } \lambda \text{ in } C\}$$

So we can identify a compatible observation with the set of edges $E_\Lambda = \bigcup_{w \in W} E_\Lambda(w) \subset E(C)$. We note that the set of compatible observations is large and in particular when the CEG is expressible as a BN contains all sets of the form $O(\mathbf{A})$ defined above.

Example 2

Consider:

$$\Lambda = \{\lambda \mid e_\lambda \in \{e_1(w_0, w_1), e_2(w_0, w_2), e_4(w_1, w_\infty), e_5(w_1, w_4), e_6(w_2, w_4), e_7(w_2, w_5), e_{10}(w_4, w_\infty), e_{11}(w_4, w_\infty), e_{14}(w_5, w_6), e_{15}(w_6, w_\infty)\}\}$$

This corresponds to all the root-to-sink paths in the subgraph of C given in Figure 3.

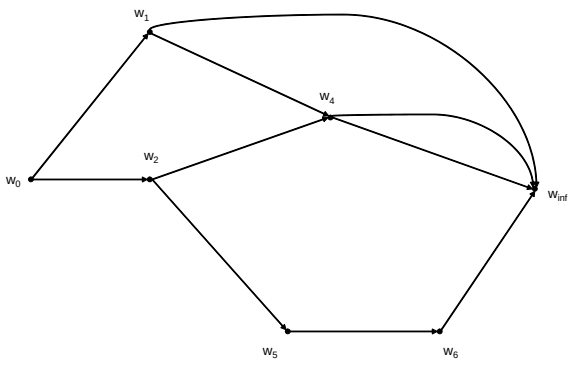


Figure 3: Subgraph for event Λ in Example 2

3.3 MESSAGE PASSING FROM COMPATIBLE OBSERVATIONS ON A CEG

The message passing algorithm is a function from the original probabilities $\{\boldsymbol{\pi}(w) \mid w \in W\}$ to revised probabilities on the same graph $\{\hat{\boldsymbol{\pi}}(w) \mid w \in W\}$ conditional on the observation Λ. Note that once edge-probabilities have been revised, the resulting graph may not be a *minimal* CEG (in that we may have vertices within the graph which are the roots of identical sub-graphs). It is possible (although unnecessary for information-propagating purposes) to add a further algorithm step to produce a minimal CEG if this is required. This step ensures that any vertices that are equivalent are combined into a single position.

Messages are passed from the terminal edges backwards through the transporter CEG along neighbouring edges until reaching the root in a *collect* step giving a new pair $\{\boldsymbol{\tau}(w), \Phi(w) \mid w \in W\}$. We then move forward from the root producing revised $\{\hat{\boldsymbol{\pi}}(w) \mid w \in W\}$. Let $W(-1)$ denote the set of positions all of whose outgoing edges terminate in w_∞ in C.

1. For any edge $e(w, w_\infty)$ such that $w \in W(-1)$, set the *potential* $\tau_e(w_\infty \mid w) = 0$ if $e(w, w_\infty) \notin E_\Lambda$, and $\tau_e(w_\infty \mid w) = \pi_e(w_\infty \mid w)$ if $e(w, w_\infty) \in E_\Lambda$. Let the *emphasis*:
$$\Phi(w) = \sum_{e \in E(w)} \tau_e(w_\infty \mid w)$$
Say that w_∞ and each of these positions is *accommodated.*
2. For any position w all of whose children are accommodated, and edge $e(w, w')$, set the *potential* $\tau_e(w' \mid w) = 0$ if $e(w, w') \notin E_\Lambda$, and $\tau_e(w' \mid w) = \pi_e(w' \mid w)\ \Phi(w')$ if $e(w, w') \in E_\Lambda$. Let the *emphasis*:
$$\Phi(w) = \sum_{e \in E(w)} \tau_e(w' \mid w)$$
Say that w is acommodated.
3. Repeat step 2 until all $w \in W$ are accommodated. This completes the collect steps.
4. For all $w \in W$, set:
$$\hat{\boldsymbol{\pi}}(w) = \mathbf{0} \quad \text{if } \boldsymbol{\tau}(w) = \mathbf{0}$$
$$\hat{\boldsymbol{\pi}}(w) = \frac{\boldsymbol{\tau}(w)}{\Phi(w)} \quad \text{if } \boldsymbol{\tau}(w) \neq \mathbf{0}$$
where $\boldsymbol{\tau}(w) = \{\tau_e(w' \mid w) \mid e(w, w') \in E(w)\}$.

Clearly we have that:

$$\hat{\pi}_e(w' \mid w) = 0 \quad \text{if } e(w, w') \notin E_\Lambda$$
$$\hat{\pi}_e(w' \mid w) = \frac{\tau_e(w' \mid w)}{\Phi(w)} \quad \text{if } e(w, w') \in E_\Lambda$$

A proof of these results is given in the appendix.

Note that as we move forward through the graph the updated probabilities of $\mu(w_0, w)$ subpaths will be of the form:

$$\hat{\pi}_\mu(w \mid w_0) = \prod_{i=0} \hat{\pi}_e(w_{i+1} \mid w_i)$$

and we get:

$$\hat{\pi}(\Lambda(w)) = \sum_{\mu \in \{\mu(w_0, w)\}} \hat{\pi}_\mu(w \mid w_0)$$

From the definition of accommodation, the order of these operations (like the perfect order used to update a triangulated BN) depends only on the toplogy of C, so it can be set up beforehand.

Example 3

Steps 1, 2 and 3 give us the graph in Figure 4. Step 4 gives us the CEG in Figure 5 (note that our CEG is again simple, and also minimal without the need for the additional step previously mentioned).

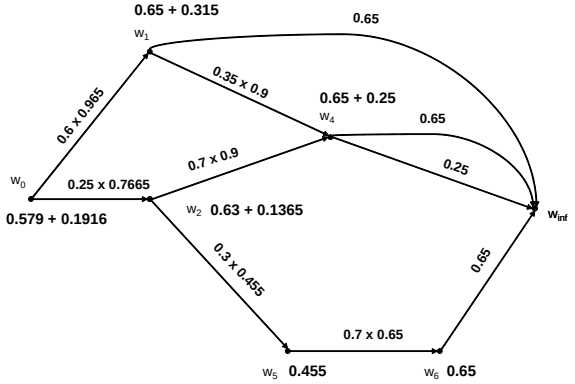


Figure 4: Potentials and emphases added

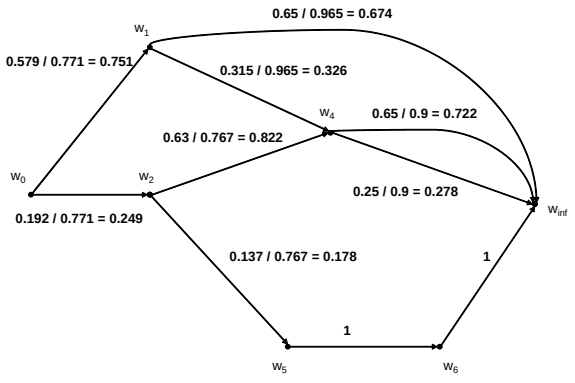


Figure 5: Updated CEG C_Λ

Note that, in analogy with equation (6) of Cowell and Dawid (1992), the conditional probability of any atom $\lambda = (w_0 = w_\lambda[0], e_\lambda[1], w_\lambda[1], \ldots, e_\lambda[n(\lambda)], w_\infty)$ is given by the invariance formula:

$$\pi(\lambda \mid \Lambda) = \hat{\pi}(\lambda) = \prod_{i=1}^{n(\lambda)} \hat{\pi}(e_\lambda[i]) = \frac{\prod_{i=1}^{n(\lambda)} \tau(e_\lambda[i])}{\prod_{i=0}^{n(\lambda)-1} \Phi(w_\lambda[i])}$$

Also note that at the cost of some computation, we can perform inference on the reduced graph C_Λ whose edges $E(C_\Lambda)$ are just the edges e in $E(C)$ with non-zero probabilities $\hat{\pi}(e)$, and whose vertices $W(C_\Lambda)$ are the $w \in W(C)$ for which $\Phi(w) \neq 0$. The non-zero edge and vertex probabilities of C then simply map on to their corresponding edge and vertex probabilities in C_Λ. Note that, unlike for the BN, any non trivial $C-$compatible observation strictly reduces the number of edges in the edge set after this operation.

A pseudo-code version of our algorithm is provided below:

Let $C(W(C), E(C))$ be a transporter CEG with edges in $E(C)$ having labels e_i, $i = 1, 2, \ldots n_e$, such that $i < j \Rightarrow e_i \not\succ e_j$ (e_i does not lie downstream of e_j on any $w_0 \to w_\infty$ path); and positions in $W(C) \backslash \{w_\infty\}$ having labels w_i, $i = 0, 1, 2, \ldots m_w$, such that $i < j \Rightarrow w_i \not\succ w_j$. To update the edge-probabilities on C following observation of an event Λ, do:

(1) Set $A = \phi$
(2) Set $B = \phi$
(3) Set $i = 1$
(4) **Repeat**
  (a) Select e_i
  (b) **If** $e_i \in E_\Lambda$, add e_i to A
    otherwise, set $\hat{\pi}_{e_i} = 0$
  (c) Set $i = i + 1$
  Until $i = n_e + 1$
(5) Set $\Phi(w_\infty) = 1$
(6) Set $j = m_w$
(7) **Repeat**
  (a) Select w_j
  (b) **Repeat**
    (i) Select $e(w_j, w'_j) \in E(w_j) \cap A$
    (ii) Set $\tau_e(w'_j \mid w_j) = \pi_e(w'_j \mid w_j)\ \Phi(w'_j)$
    (iii) Add $e(w_j, w'_j)$ to B
    Until $E(w_j) \cap A \subset B$
  (c) Set $\Phi(w_j) = \sum_{e \in E(w_j)} \tau_e(w'_j \mid w_j)$
  (d) Set $j = j - 1$
  Until $j = -1$
(8) For each $e(w, w') \in E_\Lambda$, set $\hat{\pi}_e(w' \mid w) = \frac{\tau_e(w' \mid w)}{\Phi(w)}$
(9) Return $\{\hat{\pi}_e\}$

4 A CLOSER LOOK AT OUR EXAMPLE

Consider the CEG in Figure 2 and let the 16 edges be labelled e_i in the same order as the $\{\theta_i\}$ thereon. In

Examples 1 to 3 we showed how to create and use a Transporter CEG without concerning ourselves with a context. We now add that context and suppose that this CEG represents a Treatment regime for a serious medical condition, and the edges carry the meanings given in Table 2:

Table 2: Edge descriptors

Edge	Description
e_1	Not critical – Treatment prescribed I
e_2	Liver failure – Treatment ... II
e_3	Liver & Kidney failure – Treatment ... II
e_4	Responds to I – Full recovery
e_5	No response to I – Surgery prescribed III
e_6, e_8	Responds to II – Surgery ... III
e_7, e_9	No response to II – Surgery ... IV
e_{10}	Recovery – Lifetime monitoring
e_{11}	Recovery – Lifetime medication
e_{12}, e_{13}	Death in surgery
e_{14}	Survives surgery IV – Treatment ... V
e_{15}	Recovery – Lifetime on treatment V
e_{16}	No response to V – Dies

As alluded to in section 2, it is not possible to represent this regime efficiently as a BN, nor yet as a context-specific BN, given that the asymmetry of the problem does not just lie in it having asymmetric sample space structures. By equating the descriptions of edges e_4 and e_{10}; edges e_{11} and e_{15}; and edges e_{12}, e_{13} and e_{16}, we can however **approximate** the problem with a 4-variable BN; where X_1 *Diagnosis and initial treatment* can take values corresponding to the outcomes {*Not critical, Liver failure, Liver & Kidney failure*}; X_2 2*nd treatment* to {*None,* III, IV}; X_3 3*rd treatment* to {*None,* V}; and X_4 *Response* to {*Death, Partial recovery, Full recovery*}. The BN for this approximation to the model is given in Figure 6.

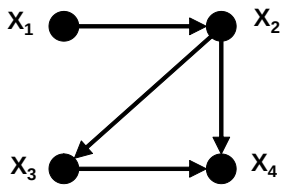


Figure 6: BN for our example

To store the model using a CEG requires 16 cells (corresponding to the 16 edges), but in this BN 27 cells (9 for the clique $\{X_1, X_2\}$ and 18 for $\{X_2, X_3, X_4\}$), 14 of which are storing the value zero.

The event Λ in our example corresponds to the observation that a patient was not diagnosed with Liver **and** Kidney failure, and is still alive. Propagation of this event enables a practitioner to establish probability distributions for the possible histories of our patient. Note that it is only the fact that we can describe Λ in such a simple manner that has allowed us to approximate the problem with the BN in Figure 6.

Propagating of the event Λ using a simple Junction Tree algorithm on the cliques of the BN takes a minimum of 43 operations. Propagation on the CEG using our algorithm requires 32 operations (corresponding to 16 *backward* edges, 6 *backward* vertices and 10 *forward* edges). So even in this simple example, using the CEG is more efficient than the BN. The efficiency here is due mainly to the fact that the clique probability tables contain many zeros. This is reflected in the CEG by the $w_0 \to w_\infty$ paths not all having the same length. It is this form of asymmetry in a model that context-specific BNs do not cope with adequately, and why CEGs are a better structure for use with this type of problem.

The problems in which the algorithm described above are most efficient are when the CEG structure is known to be *simple*. To store the probability tables for the CEG requires only $N = \#(W(C)) + \#(E(C)) < 2\#(E(C))$ cells. In this case the *collect* step involves only N calculations and the topology of the CEG is valid so that in particular the original probability table structure can be preserved. The potential product necessitates only a single distribute step which again only involves at most N calculations. For large trees with much of the type of subtree symmetry discussed above the propagation is extremely fast.

It is worth quickly looking at a very simple example arising from model selection in graphical or partition model problems, an area currently attracting some interest: Consider a model with random variables $X_1, \ldots X_n$, where X_1 with $M = {}^1/_2(n-1)(n-2)$ possible states, determines which pair of binary variables from $\{X_2, \ldots X_n\}$ are dependent, all other variables from $\{X_2, \ldots X_n\}$ being independent of each other and of the pair determined. The CEG of this model has at most $M(1+2n)$ edges and $2+Mn$ positions, whereas the BN is a single clique requiring $M \times 2^{n-1}$ cells for storage. As the number of operations required for propagation on both the BN and the CEG is of the same order of magnitude as the number of cells required for storage, it is clear that the CEG is far more efficient in this example.

5 DISCUSSION

There are several advantages of this method over the coding of this type of problem through a BN. Firstly, the calculated probabilities can be projected back on to the edges of the elicited tree, so that the consequences of inferences given different types of information can be immediately appreciated by the practi-

tioner. Secondly, the accommmodation of data in the form of a compatible observation is much more general than the accommodation of subsets of observations from a predetermined set of random variables, so the CEG provides a more flexible framework for propagation, particularly when data is contingently censored. Thirdly, there are efficiency gains as outlined above. We intend to show how great these gains can be for very large problems in a later paper.

Note also that, as is the case with the triangulation step in BN-based algorithms, there are faster algorithms (Thwaites 2008) than the one described above, although they lose some of this algorithm's generality. Our algorithms are currently being coded by Cowell within freely available software, and will be available shortly.

Of course BNs provide a simpler representation of more symmetric problems and should always be preferred when the three contingencies are not satisified. The CEG does not provide a universal improvement over the BN for propagation. In particular in problems when the underlying BN is decomposable but the CEG is not simple the BN propagation can be much more efficient. But in highly asymmetric problems, the CEG should definitely be a first choice.

It should be noted that it is also possible to define a *dynamic* analogue of the CEG, and our investigation of these suggests that a *time-sliced* CEG (analogous to a *time-sliced* BN) will be an ideal vehicle for a dynamic updating algorithm. We hope to report on these developments in the near-future.

APPENDIX

We claim that:

$$\begin{aligned}\hat{\pi}_e(w' \mid w) &\triangleq \pi(\Lambda(e(w,w')) \mid \Lambda, \Lambda(w)) \\ &= \begin{cases} \frac{\tau_e(w' \mid w)}{\Phi(w)} & \text{if } e(w,w') \in E_\Lambda \\ 0 & \text{if } e(w,w') \notin E_\Lambda \end{cases}\end{aligned}$$

Proof:

For a CEG C, and $C-$compatible event Λ, let T be the tree associated with C, T_Λ the tree associated with C_Λ, and $T_{(\Lambda)}$ the subtree of T containing only those root-to-leaf paths in Λ. $T_{(\Lambda)}$ differs from T_Λ in that the former retains the edge-probabilities from T.

Consider a position $w \in C$ ($w \in C_\Lambda$) corresponding to a set of vertices $\{v_i\} \in T$. Then the subtrees rooted in each v_i are identical both in topology and in their edge-probabilities.

If there is a subpath $\mu(w_0, w)$ which is **not** part of a $w_0 \to w_\infty$ path in Λ (ie. $\mu(w_0, w)$ exists in C, but **not** in C_Λ) then there will exist a subset of $\{v_i\}$ which does not exist in T_Λ (or $T_{(\Lambda)}$). We split the set $\{v_i\}$ into:

$\{v_i\}_{i\in I}$ vertices existing in T_Λ
$\{v_i\}_{i\in J}$ vertices **not** existing in T_Λ

Because Λ is $C-$compatible, the subtrees in $T_{(\Lambda)}$ rooted in each $v_i \in \{v_i\}_{i\in I}$ are also identical both in topology and in their edge-probabilities that they retain from T.

Suppose there exists an edge $e(w,w')$ in C, then for each $v_i \in \{v_i\}$, there exists an edge $e(v_i, v_i')$ in T corresponding to this edge. Note that:

$$\Lambda(w) = \bigcup_{i\in I\cup J} \Lambda(v_i)$$

$$\Lambda(e(w,w')) = \bigcup_{i\in I\cup J} \Lambda(e(v_i, v_i'))$$

$$\pi_e(v_i' \mid v_i) = \pi_e(w' \mid w) \quad \forall i \in I \cup J$$

and since the subtrees in $T_{(\Lambda)}$ rooted in each $v_i \in \{v_i\}_{i\in I}$ are identical, we also have:

$$\pi(\Lambda \mid \Lambda(v_i)) = \pi(\Lambda \mid \Lambda(v_j))$$

$$\pi(\Lambda, \Lambda(e(v_i, v_i')) \mid \Lambda(v_i)) = \pi(\Lambda, \Lambda(e(v_j, v_j')) \mid \Lambda(v_j)) \quad \text{for } i,j \in I$$

[$\pi(\Lambda \mid \Lambda(v_i))$ is the sum of the probabilities of all the $\mu(v_i, v_{leaf})$ subpaths in $T_{(\Lambda)}$, and $\pi(\Lambda, \Lambda(e(v_i, v_i')) \mid \Lambda(v_i))$ is the sum of the probabilities of all the $\mu(v_i, e(v_i, v_i'), v_i', v_{leaf})$ subpaths in $T_{(\Lambda)}$]

So:

$$\begin{aligned}\hat{\pi}_e(w' \mid w) &= \pi(\Lambda(e(w,w')) \mid \Lambda, \Lambda(w)) \\ &= \frac{\pi(\Lambda, \Lambda(w), \Lambda(e(w,w')))}{\pi(\Lambda, \Lambda(w))} \\ &= \frac{\pi(\Lambda, \bigcup_{i\in I\cup J}[\Lambda(v_i), \Lambda(e(v_i, v_i'))])}{\pi(\Lambda, \bigcup_{i\in I\cup J} \Lambda(v_i))}\end{aligned}$$

(an expression evaluated on T)

since $\Lambda(v_i) \cap \Lambda(e(v_j, v_j')) = \phi$ for $i \neq j$

$$= \frac{\sum_{i\in I\cup J} \pi(\Lambda, \Lambda(v_i), \Lambda(e(v_i, v_i')))}{\sum_{i\in I\cup J} \pi(\Lambda, \Lambda(v_i))}$$

But $\Lambda \cap \Lambda(v_i) = \phi$ for $v_i \in \{v_i\}_{i\in J}$, so this equals:

$$\begin{aligned}&\frac{\sum_{i\in I} \pi(\Lambda, \Lambda(v_i), \Lambda(e(v_i, v_i')))}{\sum_{i\in I} \pi(\Lambda, \Lambda(v_i))} \\ &= \frac{\sum_{i\in I} \pi(\Lambda, \Lambda(e(v_i, v_i')) \mid \Lambda(v_i))\ \pi(\Lambda(v_i))}{\sum_{i\in I} \pi(\Lambda \mid \Lambda(v_i))\ \pi(\Lambda(v_i))} \\ &= \frac{\pi(\Lambda, \Lambda(e(v_j, v_j')) \mid \Lambda(v_j))\ \sum_{i\in I} \pi(\Lambda(v_i))}{\pi(\Lambda \mid \Lambda(v_j))\ \sum_{i\in I} \pi(\Lambda(v_i))}\end{aligned}$$

for any $v_j \in \{v_i\}_{i\in I}$

$$= \frac{\pi(\Lambda, \Lambda(e(v_j, v_j')) \mid \Lambda(v_j))}{\pi(\Lambda \mid \Lambda(v_j))}$$

for any $v_j \in \{v_i\}_{i\in I}$

Turning our attention to the terms in the algorithm, we claim that $\Phi(w) = \pi(\Lambda \mid \Lambda(v_i))$ and $\tau_e(w' \mid w) = \pi(\Lambda, \Lambda(e(v_i, v_i')) \mid \Lambda(v_i))$ ($v_i \in \{v_i\}_{i\in I}$) for all w, $e(w,w') \in C_\Lambda$, where $\{v_i\}_{i\in I}$ is the set of vertices in $T_{(\Lambda)}$ corresponding to w. We prove this by induction:

Step 1.

Consider positions $w \in W(-1)$. Then:

$$\Phi(w) = \sum_e \tau_e(w_\infty \mid w) = \sum_e \pi_e(w_\infty \mid w)$$
$$= \sum_e \pi_e(v_{leaf} \mid v_i) \qquad \text{in } T_{(\Lambda)}$$

for any $v_i \in \{v_i\}_{i \in I}$

$$= \pi(\Lambda \mid \Lambda(v_i))$$

Step 2.

Suppose w is such that all of its outgoing edges terminate in positions $\{w'\}$ for which $\Phi(w') = \pi(\Lambda \mid \Lambda(v_i'))$. Then:

$$\Phi(w) = \sum_e \tau_e(w' \mid w) = \sum_e \pi_e(w' \mid w)\ \Phi(w')$$
$$= \sum_e \pi_e(v_i' \mid v_i)\ \pi(\Lambda \mid \Lambda(v_i'))$$

for any $v_i \in \{v_i\}_{i \in I}$

$$= \sum_e \pi(\Lambda(e(v_i, v_i')) \mid \Lambda(v_i))\ \pi(\Lambda \mid \Lambda(v_i'))$$

But $\Lambda(v_i') = \Lambda(e(v_i, v_i')) \subset \Lambda(v_i)$ in a tree, so this equals:

$$\sum_e \pi(\Lambda(e(v_i, v_i')), \Lambda(v_i') \mid \Lambda(v_i))$$
$$\times\ \pi(\Lambda \mid \Lambda(v_i), \Lambda(e(v_i, v_i')), \Lambda(v_i'))$$
$$= \sum_e \pi(\Lambda, \Lambda(e(v_i, v_i')), \Lambda(v_i') \mid \Lambda(v_i))$$
$$= \sum_e \pi(\Lambda, \Lambda(e(v_i, v_i')) \mid \Lambda(v_i))$$
$$= \pi(\Lambda, \Lambda(v_i) \mid \Lambda(v_i)) = \pi(\Lambda \mid \Lambda(v_i))$$

Hence:

$$\tau_e(w' \mid w) = \pi_e(w' \mid w)\ \Phi(w')$$
$$= \pi_e(v_i' \mid v_i)\ \pi(\Lambda \mid \Lambda(v_i'))$$

for any $v_i \in \{v_i\}_{i \in I}$

$$= \pi(\Lambda(e(v_i, v_i')) \mid \Lambda(v_i))\ \pi(\Lambda \mid \Lambda(v_i'))$$
$$= \ldots = \pi(\Lambda, \Lambda(e(v_i, v_i')) \mid \Lambda(v_i))$$

We now combine our two results to give:

$$\hat{\pi}_e(w' \mid w) = \frac{\pi(\Lambda, \Lambda(e(v_j, v_j')) \mid \Lambda(v_j))}{\pi(\Lambda \mid \Lambda(v_j))}$$
$$= \frac{\tau_e(w' \mid w)}{\Phi(w)} \qquad \square$$

Acknowledgements

This work has been partly funded by the EPSRC as part of the project *Chain Event Graphs: Semantics and Inference.*

References

[1] P. E. Anderson and J. Q. Smith. Conditional independence and Chain Event Graphs. *Artificial Intelligence*, 172:42–68, 2008.

[2] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian Networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 115–123, Portland, Oregon, 1996.

[3] R. G. Cowell and A. P. Dawid. Fast retraction of evidence in a probabilistic expert system. *Statistics and Computing*, 2:37–40, 1992.

[4] S. French and D. R. Insua. *Statistical Decision Theory.* Arnold, 2000.

[5] D. McAllester, M. Collins, and F. Periera. Case factor diagrams for structured probabilistic modeling. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 382–391, 2004.

[6] D. Poole and N. L. Zhang. Exploiting contextual independence in probabilistic inference. *Journal of Artificial Intelligence Research*, 18:263–313, 2003.

[7] E. M. Riccomagno and J. Q. Smith. The causal manipulation and Bayesian estimation of Chain Event Graphs. Research Report 05-16, CRiSM, 2005.

[8] E. M. Riccomagno and J. Q. Smith. The geometry of causal probability trees that are algebraically constrained. In L. Pronzato and A. Zhigljavsky, editors, *Optimal Design and Related Areas in Optimization and Statistics*, chapter 6, pages 131–151. Springer-Verlag, 2007.

[9] A. Salmeron, A. Cano, and S. Moral. Importance sampling in Bayesian Networks using probability trees. *Computational Statistics and Data Analysis*, 34:387–413, 2000.

[10] G. Shafer. *The Art of Causal Conjecture.* MIT Press, 1996.

[11] P. A. Thwaites. *Chain Event Graphs: Theory and application.* PhD thesis, University of Warwick, 2008.

[12] P. A. Thwaites and J. Q. Smith. Evaluating causal effects using Chain Event Graphs. In *Proceedings of the 3rd European Workshop on Probabilistic Graphical Models*, pages 291–300, Prague, 2006.

Identifying Dynamic Sequential Plans

Jin Tian
Department of Computer Science
Iowa State University
Ames, IA 50011
jtian@cs.iastate.edu

Abstract

We address the problem of identifying dynamic sequential plans in the framework of causal Bayesian networks, and show that the problem is reduced to identifying causal effects, for which there are complete identification algorithms available in the literature.

1 Introduction

This paper deals with the problem of evaluating the effects of sequential plans from a combination of nonexperimental data and qualitative causal assumptions. The causal assumptions will be represented in the form of an acyclic causal diagram [Spirtes *et al.*, 1993, Heckerman and Shachter, 1995, Lauritzen, 2000, Pearl, 2000] in which arrows represent the potential existence of direct causal relationships between the corresponding variables. The causal diagram may contain unmeasured variables, and our task is to decide whether we can estimate the effects of a sequence of actions from the observed data.

We motivate the study by considering a medical treatment problem discussed in [Pearl and Robins, 1995, Dawid and Didelez, 2005]. There are a sequence of medical treatments $(X_1, \ldots, X_k)$ applied to a patient over time. We have observations Z_i before and between the treatments. Doctors may prescribe a treatment based on previous treatments and observations. There is a outcome variable Y (say survival) of special interest and we want to estimate the effects of the sequential treatments on Y. In general there may be unobserved confounders that have influence on the observed variables.

There are different possible strategies for choosing the treatment action (X_i). The simplest action involves fixing the value of X_i to a particular value x_i^*, called atomic intervention and denoted by $do(x_i^*)$ in [Pearl, 2000], e.g. fixing the dosage of a treatment irrespective of any observations on the patient. A unconditional plan consists of a sequence of pre-defined atomic actions. The problem of identifying unconditional plans is to compute the distribution of Y under atomic interventions on a set X of action variables, denoted by $P_x(y) = P(y|do(x))$, a quantity known as the causal effects of X on Y. Sufficient graphical criteria for the identifiability of unconditional plans are derived in [Pearl and Robins, 1995]. Recently the general problem of identifying causal effects $P_x(y)$ in a causal diagram containing unobserved variables has been solved and complete algorithms for identification are given in [Tian and Pearl, 2003, Shpitser and Pearl, 2006b, Huang and Valtorta, 2006].

In general we may want to use dynamic treatment strategies in which the values of action variables (X_i) are determined based on the previously observed variables and previously taken actions. Sufficient graphical criteria for the identifiability of dynamic plans are derived in [Dawid and Didelez, 2005]. However the identifiability problem is far from being solved.

In this paper, we show that the problem of identifying dynamic sequential plans can be reduced to the well-studied problem of identifying causal effects and therefore essentially solved the sequential plan identification problem. Although Pearl (2000, Section 4.2) has suggested that dynamic conditional plans may be identified by identifying conditional causal effects of the form $P_x(y|c)$, for which complete identification algorithms have been developed [Tian, 2004, Shpitser and Pearl, 2006a], in this paper, we will show that this gives a sufficient condition for identifying dynamic sequential plans but it is not necessary.

The rest of the paper is organized as follows. In Section 2, we review the work in [Dawid and Didelez, 2005] and define useful notation. In Section 3, we formulate the sequential plan problem in the framework of causal Bayesian networks. We

show how to reduce the problem of identifying dynamic sequential plans into a problem of identifying causal effects in Section 4, and discuss in Section 5 the problem versus that of identifying unconditional plans and conditional causal effects. Section 6 concludes the paper.

2 Previous Work and Notation

Dawid and Didelez (2005) formulated the problem of identifying dynamic sequential plans in the framework of regime indicators and influence diagrams. An influence diagram (ID) is a DAG over a set $V = \{V_1, \ldots, V_n\}$ of variables that also includes regime indicators as special nodes of their own called decision nodes [Dawid, 2002]. We assume that all variables are discrete. The DAG is assumed to represent conditional independence assertions that each variable is independent of all its non-descendants given its direct parents in the graph.[1] These assertions imply that the joint probability function $P(v) = P(v_1, \ldots, v_n)$ factorizes according to the product [Pearl, 1988]

$$P(v) = \prod_i P(v_i|pa_i) \tag{1}$$

where pa_i are (values of) the parents of variable V_i in the graph.[2]

The question of causal inference is considered as a problem of inference across different *regimes*, in which we may intervene in certain variables in certain ways and observe the behavior of other variables. *Regime indicators* are used to represent different types of interventions. Here we will roughly follow the notation used in [Didelez *et al.*, 2006]. The regime indicator for an intervention in a variable V_i is denoted by σ_{V_i} and can take values in a set of strategies. Under strategy σ_{V_i}, the conditional probability $P(v_i|pa_i)$ is changed to $P(v_i|pa_i; \sigma_{V_i})$. We will consider the following types of interventions.

- *Idle regime* $\sigma_{V_i} = \emptyset$*:* No intervention takes place, therefore

  $$P(v_i|pa_i; \sigma_{V_i} = \emptyset) = P(v_i|pa_i).$$

  The idle regime is also called the observational regime under which we will assume that observational data has been collected. Therefore $P(v_i|pa_i)$ can be estimated from data if V_i and Pa_i are observed.

- *Atomic intervention* $\sigma_{V_i} = do(v_i^*)$*:* The strategy of setting V_i to a fixed value v_i^*, denoted by $do(V_i = v_i^*)$ or simply $do(v_i^*)$ in Pearl (2000), such that

  $$P(v_i|pa_i; \sigma_{V_i} = do(v_i^*)) = \delta(v_i, v_i^*),$$

  where $\delta(v_i, v_i^*)$ is one if $v_i = v_i^*$ and zero otherwise.

- *Conditional intervention* $\sigma_{V_i} = do(g(c))$*:* In general, V_i may be made to respond in a specified way to some set C of previously observed variables, denoted by $do(V_i = g(c))$ in Pearl (2000), such that

  $$P(v_i|pa_i; \sigma_{V_i} = do(g(c))) = \delta(v_i, g(c)),$$

  where $g(.)$ is a pre-specified deterministic function and the variables in C can not be descendants of V_i.

- *Random intervention* $\sigma_{V_i} = d_C$*:* More generally, we may let V_i take on a random value according to some distribution possibly depending on some set C of previously observed variables such that

  $$P(v_i|pa_i; \sigma_{V_i} = d_C) = P^*(v_i|c),$$

  where $P^*(v_i|c)$ is a pre-specified probability distribution and the variables in C can not be descendants of V_i.

In a sequential decision problem, we may intervene, at least in principle, in a set of variables $X = \{X_i\} \subset V$, called *control variables* or *action variables*, and are interested in the response of a variable Y, called *response variable* or *outcome variable*. Let Z be the rest of observed variables which are often called *covariates*. The variables are assumed to be ordered in a sequence $(L_1, X_1, \ldots, L_K, X_K, Y)$ where $L_i \subseteq Z$ are the set of observed covariates after X_{i-1} and before X_i. We denote $\bar{L}_i = (L_1, \ldots, L_i)$ and $\bar{X}_i = (X_1, \ldots, X_i)$.

Given an intervention strategy $\sigma_X = \{\sigma_{X_i}\}$, under a condition called *simple stability* which says that the observed covariates L_i and the outcome Y are independent of how action variables are generated once all earlier observables ($\bar{L}_{i-1}, \bar{X}_{i-1}$ for L_i; X, Z for Y) are given, Dawid and Didelez (2005) show that the post-intervention distribution of Y is identified as

$$P(y; \sigma_X) = \sum_{x,z} P(y|x,z) \prod_i P(l_i|\bar{l}_{i-1}, \bar{x}_{i-1}) \prod_i P(x_i|\bar{x}_{i-1}, \bar{l}_i; \sigma_{X_i}), \tag{2}$$

where $P(x_i|\bar{x}_{i-1}, \bar{l}_i; \sigma_{X_i})$ are determined by the chosen regime and the other quantities can be estimated from

[1] We use family relationships such as "parents," "children," and "ancestors" to describe the obvious graphical relationships.

[2] We use uppercase letters to represent variables or sets of variables, and use corresponding lowercase letters to represent their values (instantiations).

observational data. Eq. (2) is known as the *G-formula*, and has been obtained in [Robins, 1986, Robins, 1987] in the framework of potential response models.

When there are unobserved confounders, the simple stability may not hold. Dawid and Didelez (2005) makes *extended stability* assumption which essentially is (simple) stability with respect to the extended domain that includes unobserved U variables ignoring the distinction between Z and U. The G-formula (2) no longer holds unless we include unobserved U variables, but then the conditional probabilities involving U variables can no longer be estimated from the data. Sufficient graphical criteria for identifying $P(y; \sigma_X)$ are derived. The criteria were obtained by identifying graphical conditions under which the simple stability can be regained such that the G-formula can be used, and by extending the work in [Pearl and Robins, 1995] to dynamic plans.

3 Problem Formulation

In this paper, we will formulate the sequential plan problem in the framework of causal Bayesian networks. A *causal Bayesian network (CBN)* consists of a DAG G over a set $V = \{V_1, \ldots, V_n\}$ of variables, called a *causal diagram*. The interpretation of such a graph has two components, probabilistic and causal. The probabilistic interpretation views G as representing conditional independence assertions such that the joint probability function $P(v) = P(v_1, \ldots, v_n)$ factorizes according to Eq. (1). The causal interpretation views the directed edges in G as representing causal influences between the corresponding variables. In this interpretation, the factorization of (1) still holds, but the factors are further assumed to represent autonomous data-generation processes, that is, each conditional probability $P(v_i|pa_i)$ represents a stochastic process by which the values of V_i are chosen in response to the values pa_i (previously chosen for V_i's parents), and the stochastic variation of this assignment is assumed independent of the variations in all other assignments. Moreover, each assignment process remains invariant to possible changes in the assignment processes that govern other variables in the system. This modularity assumption enables us to predict the effects of interventions, whenever interventions are described as specific modifications of some factors in the product of (1). We typically assume that every variable V_i can potentially be manipulated by external intervention. So we might think of a CBN as an ID such that each node is (implicitly) pointed to by a corresponding regime/intervention indicator.

In a sequential decision problem, we may intervene in a set of action variables $X = \{X_i\} \subset V$, and are interested in the response of a set of outcome variables Y. Assume that all the variables V are observed and let the rest of covariate variables be $Z = \{Z_i\} = V \setminus (X \cup Y)$. Given an intervention strategy $\sigma_X = \{\sigma_{X_i}\}$, by modularity assumption, we can predict the effects of σ_X as

$$\begin{aligned} &P(v; \sigma_X) \\ &= \prod_i P(y_i|pa_{y_i}) \prod_i P(z_i|pa_{z_i}) \prod_i P(x_i|pa_{x_i}; \sigma_{X_i}), \end{aligned} \tag{3}$$

where, by modularity assumption, those conditional probabilities corresponding to unmanipulated variables remain unaltered. We note that Dawid and Didelez's (2005) simple stability assumption leads to Eq. (3) in the framework of CBNs. We see that, given a CBN, whenever all variables in V are observed, the consequence of an intervention strategy on the outcome variables Y is computed as

$$\begin{aligned} P(y; \sigma_X) = \sum_{x,z} &\prod_i P(y_i|pa_{y_i}) \prod_i P(z_i|pa_{z_i}) \\ &\prod_i P(x_i|pa_{x_i}; \sigma_{X_i}), \end{aligned} \tag{4}$$

where $P(x_i|pa_{x_i}; \sigma_{X_i})$ are determined by the chosen regime and the other quantities can be estimated from observational data. We note that the G-formula (2) can be reduced to Eq. (4) by using the conditional independence relationships implied by the CBN that each variable is independent of all its non-descendants given its parents.

In general we may be concerned with confounding effects due to unobserved influential variables. In the presence of unobserved confounders, the distribution over observed variables can no longer factorize according to (1). Letting $V = Y \cup Z \cup X$ and $U = \{U_1, \ldots, U_{n'}\}$ stand for the sets of observed and unobserved variables, respectively, the observed probability distribution, $P(v)$, becomes a mixture of products:

$$P(v) = \sum_u \prod_{\{i|V_i \in V\}} P(v_i|pa_{v_i}) \prod_{\{i|U_i \in U\}} P(u_i|pa_{u_i}). \tag{5}$$

We still make modularity assumption in the CBN with unobserved variables, and the effects of an intervention strategy σ_X on the outcome variables Y can be expressed as

$$\begin{aligned} P(y; \sigma_X) = \sum_{x,z,u} &\prod_i P(y_i|pa_{y_i}) \prod_i P(z_i|pa_{z_i}) \\ &\prod_i P(x_i|pa_{x_i}; \sigma_{X_i}) \prod_i P(u_i|pa_{u_i}). \end{aligned} \tag{6}$$

We note that Dawid and Didelez's (2005) extended stability assumption leads to Eq. (6) in the framework of CBNs. In (6), the quantities $P(y_i|pa_{y_i})$ and $P(z_i|pa_{z_i})$ (and $P(u_i|pa_{u_i})$) may involve elements of U and may not be estimable from data. Then the question of identifiability arises, i.e., whether it is possible to express $P(y;\sigma_X)$ as a function of the observed distribution $P(v)$.

Definition 1 [*Plan Identifiability*]
A sequential plan is said to be identifiable if $P(y;\sigma_X)$ is uniquely computable from the observed distribution $P(v)$.

4 Identification of Sequential Plans

First we make the following assumption about the type of interventions we will consider.

Assumption 1 *$P(x_i|pa_{x_i};\sigma_{X_i})$ does not depend on the unobserved variable. That is, for conditional intervention $\sigma_{X_i} = do(g(c))$ or random intervention $\sigma_{X_i} = d_C$, we require $C \subseteq X \cup Z$.*

This assumption corresponds to Condition 6.6 or 7.2 in [Dawid and Didelez, 2005].

Under Assumption 1, Eq. (6) becomes

$$P(y;\sigma_X) = \sum_{x,z} \prod_i P(x_i|pa_{x_i};\sigma_{X_i}) \sum_u \prod_i P(y_i|pa_{y_i}) \prod_i P(z_i|pa_{z_i}) \prod_i P(u_i|pa_{u_i}) \quad (7)$$

$$= \sum_{x,z} \prod_i P(x_i|pa_{x_i};\sigma_{X_i}) P_x(y,z) \quad (8)$$

Obviously a sufficient condition for $P(y;\sigma_X)$ being identifiable is that the causal effect $P_x(y,z)$ is identifiable. In particular, a simple sufficient condition for $P_x(y,z)$ being identifiable is if all the parents of action (X) variables are observables, which is Condition 6.3 in [Dawid and Didelez, 2005].

Proposition 1 *If all the parents of action (X) variables are observables, then $P(y;\sigma_X)$ is identifiable [Dawid and Didelez, 2005].*

Proof: If all the parents of action (X) variables are observables, then $P(x_i|pa_{x_i})$ contains no unobserved (U) variables, and Eq. (5) can be written as

$$P(v) = \prod_i P(x_i|pa_{x_i}) P_x(y,z), \quad (9)$$

from which we obtain that $P_x(y,z)$ is identified as

$$P_x(y,z) = \frac{p(x,y,z)}{\prod_i P(x_i|pa_{x_i})} \quad (10)$$

$$= \prod_{\{i|V_i \in Y \cup Z\}} P(v_i|\bar{v}_i), \quad (11)$$

where we have used the chain rule assuming an order of V variables that is consistent with the DAG and $\bar{v}_i$ denotes the V variables ordered ahead of V_i. Hence the sequential plan is identified as

$$P(y;\sigma_X) = \sum_{x,z} \prod_i P(x_i|pa_{x_i};\sigma_{X_i}) \prod_{\{i|V_i \in Y \cup Z\}} P(v_i|\bar{v}_i), \quad (12)$$

which is essentially the G-formula (2). □

In general $P_x(y,z)$ being identifiable is not a necessary condition for $P(y;\sigma_X)$ being identifiable. Eq. (7) may be simplified in that a factor $P(z_i|pa_{z_i})$ may be summed out (using $\sum_{z_i} P(z_i|pa_{z_i}) = 1$) if Z_i does not appear in any other factors (graphically, if Z_i does not have any children). We can derive stronger identification criterion by summing out as many factors as possible from Eq. (7). Before presenting our result, we first introduce some notation.

Following [Tian and Pearl, 2003], for any observed set $S \subseteq V$ of variables, we define the quantity $Q[S]$ to denote the post-intervention distribution of S under atomic interventions to all other variables:

$$Q[S](v) = P_{v \setminus s}(s) = \sum_u \prod_{\{i|V_i \in S\}} P(v_i|pa_{v_i}) \prod_{\{i|U_i \in U\}} P(u_i|pa_{u_i}). \quad (13)$$

For convenience, we will often write $Q[S](v)$ as $Q[S]$. Eq. (7) can be written as

$$P(y;\sigma_X) = \sum_{x,z} \prod_i P(x_i|pa_{x_i};\sigma_{X_i}) Q[Y \cup Z]. \quad (14)$$

Let G_{σ_X} denote *the manipulated graph* under the intervention strategy σ_X, which can be constructed from the original causal graph G as follows:

- For an atomic intervention $\sigma_{X_i} = do(x_i)$, cut off all the arrows entering X_i;
- For a conditional intervention $\sigma_{X_i} = do(g_i(c_i))$ or a random intervention $\sigma_{X_i} = d_{C_i}$, cut off all the arrows entering X_i and then add an arrow entering X_i from each variable in C_i.

Based on Eq. (14), we obtain the following sufficient criterion for identifying $P(y;\sigma_X)$.

Theorem 1 *Let Z_D be the set of variables in Z that are ancestors of Y in G_{σ_X}. $P(y;\sigma_X)$ is identifiable if the causal effects $Q[Y \cup Z_D] = P_{x,z\setminus z_D}(y, z_D)$ is identifiable.*

Proof: Let X_D be the set of variables in X that are ancestors of Y in G_{σ_X}. Then all the non-ancestors of Y can be summed out from Eq. (14) leading to

$$P(y;\sigma_X) = \sum_{x_D, z_D} \prod_{\{i|X_i \in X_D\}} P(x_i|pa_{x_i};\sigma_{X_i}) Q[Y \cup Z_D] \tag{15}$$

$$= \sum_{x_D, z_D} \prod_{\{i|X_i \in X_D\}} P(x_i|pa_{x_i};\sigma_{X_i}) P_{x,z\setminus z_D}(y, z_D). \tag{16}$$

□

We conjecture that the condition in Theorem 1 is also necessary. It might appear that Eq. (15) can be further simplified as follows. Let $Z_{\sigma_{X_D}}$ be the set of Z variables that appear in the term $\prod_{\{i|X_i \in X_D\}} P(x_i|pa_{x_i};\sigma_{X_i})$ (the set of conditioning variables in the strategy σ_{X_D}). Then the term $\prod_{\{i|X_i \in X_D\}} P(x_i|pa_{x_i};\sigma_{X_i})$ is a function of X_D and $Z_{\sigma_{X_D}}$. Eq. (15) becomes

$$P(y;\sigma_X)$$
$$= \sum_{x_D, z_{\sigma_{X_D}}} \prod_{\{i|X_i \in X_D\}} P(x_i|pa_{x_i};\sigma_{X_i}) \sum_{z_D \setminus z_{\sigma_{X_D}}} Q[Y \cup Z_D] \tag{17}$$

$$\equiv \sum_{x_D, z_{\sigma_{X_D}}} g(x_D, z_{\sigma_{X_D}}) \sum_{z_D \setminus z_{\sigma_{X_D}}} Q[Y \cup Z_D] \tag{18}$$

From Eq. (18), $P(y;\sigma_X)$ is identifiable if $\sum_{z_D \setminus z_{\sigma_{X_D}}} Q[Y \cup Z_D]$ is identifiable, and intuitively, the condition appears to be necessary too, since the term $g(x_D, z_{\sigma_{X_D}})$ is specified externally and no more factors can be summed out (as far as the function $g(.)$ is not independent of any variables in $z_{\sigma_{X_D}}$). A strict proof of this necessity is still under study.

On the other hand, due to the fact that none of the factors corresponding to the variables in $Z_D \setminus Z_{\sigma_{X_D}}$ can be summed out from $Q[Y \cup Z_D]$, it has been shown that $\sum_{z_D \setminus z_{\sigma_{X_D}}} Q[Y \cup Z_D]$ can be identified only via identifying $Q[Y \cup Z_D]$ and it is a if and only if condition (Lemma 11 in [Huang and Valtorta, 2006]). So from the point of view of identifying $P(y;\sigma_X)$ the reduction from Eq. (15) to Eq. (17) is not necessary.

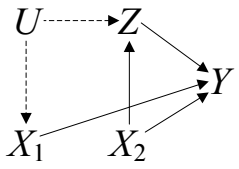


Figure 1: An example causal graph

We therefore have reduced the problem of identifying dynamic sequential plans $P(y;\sigma_X)$ into that of identifying causal effects $Q[Y \cup Z_D]$ while the latter problem has been solved and complete algorithms are given in [Tian and Pearl, 2003, Shpitser and Pearl, 2006b, Huang and Valtorta, 2006].

We demonstrate the application of Theorem 1 and the identification process with an example. Consider the problem of identifying $P(y;\sigma_{X_1},\sigma_{X_2})$ in Figure 1, which was studied in [Dawid and Didelez, 2005] and is troubling to the methods presented therein. Theorem 1 calls for identifying $Q[\{Y,Z\}]$ which can be shown to be identifiable. We are given the observational distribution

$$P(v) = P(y|x_1,x_2,z)P(x_2)Q[\{Z,X_1\}], \tag{19}$$

where

$$Q[\{Z,X_1\}] = \sum_u P(z|x_2,u)P(x_1|u)P(u). \tag{20}$$

We want to compute

$$P(y;\sigma_{X_1},\sigma_{X_2})$$
$$= \sum_{x_1,x_2,z} P(x_1;\sigma_{X_1})P(x_2;\sigma_{X_2})P(y|x_1,x_2,z)Q[\{Z\}], \tag{21}$$

which calls for computing $Q[\{Z\}]$. From Eq. (20), it is clear that

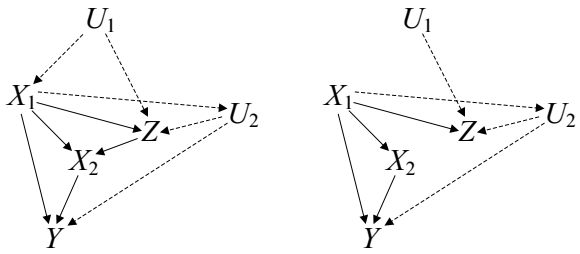
$$Q[\{Z\}] = \sum_{x_1} Q[\{Z,X_1\}]. \tag{22}$$

From Eq. (19), we obtain

$$Q[\{Z,X_1\}] = P(z,x_1|x_2) \tag{23}$$

Therefore $Q[\{Z\}]$ is identified and we finally obtain

$$P(y;\sigma_{X_1},\sigma_{X_2})$$
$$= \sum_{x_1,x_2,z} P(x_1;\sigma_{X_1})P(x_2;\sigma_{X_2})P(y|x_1,x_2,z)P(z|x_2). \tag{24}$$



(a) G (b) G_{σ_X}

Figure 2: An example causal graph

5 Discussion

5.1 Unconditional plans are easier

In general identifying dynamic plans is more difficult than identifying unconditional plans that involve only atomic interventions. Let an intervention strategy σ'_X consist of all atomic interventions. Then the manipulated graph $G_{\sigma'_X}$ is a subgraph of G_{σ_X}. Let Z'_D be the set of variables in Z that are ancestors of Y in $G_{\sigma'_X}$. Then Z'_D is a subset of Z_D. We have

$$P(y;\sigma'_X) = P_x(y) = \sum_{z'_D} Q[Y \cup Z'_D]. \quad (25)$$

Therefore identifying $P_x(y)$ calls for identifying $Q[Y \cup Z'_D]$ while $P(y;\sigma_X)$ calls for identifying $Q[Y \cup Z_D]$. Now we have

$$Q[Y \cup Z'_D] = \sum_{z_D \setminus z'_D} Q[Y \cup Z_D] \quad (26)$$

The factors of the variables in $Z_D \setminus Z'_D$ are summed out from $Q[Y \cup Z_D]$ since the variables in $Z_D \setminus Z'_D$ can be ancestors of Y only through X_i's. In general, whenever $Q[Y \cup Z_D]$ (and therefore $P(y;\sigma_X)$) is identifiable, then $P_x(y)$ is identifiable. However, it is possible that $P_x(y)$ is identifiable but $Q[Y \cup Z_D]$ (and therefore $P(y;\sigma_X)$) is not.

We demonstrate this point with an example. Consider the problem of identifying $P(y;\sigma_{X_1},\sigma_{X_2})$ in Figure 2(a), which was studied in [Pearl and Robins, 1995]. If $P(x_2|x_1,z;\sigma_{X_2})$ depends on Z, say $\sigma_{X_2} = do(g(x_1,z))$, then $Z_D = \{Z\}$, and by Theorem 1, to identify $P(y;\sigma_{X_1},\sigma_{X_2})$ we need to identify $Q[\{Y,Z\}]$, which can be shown to be not identifiable (by theorems in [Huang and Valtorta, 2006]). More specifically, given the observational distribution

$$P(v) = P(x_2|x_1,z)Q[\{X_1,Z,Y\}], \quad (27)$$

we want to identify

$$P(y;\sigma_{X_1},\sigma_{X_2}) = \sum_{x_1,x_2,z} P(x_1;\sigma_{X_1})P(x_2|x_1,z;\sigma_{X_2})Q[\{Y,Z\}] \quad (28)$$

From Eq. (28), we see that if $P(x_2|x_1,z;\sigma_{X_2})$ depends on Z, then the identifiability of $P(y;\sigma_{X_1},\sigma_{X_2})$ depends on the identifiability of $Q[\{Y,Z\}]$. We therefore conclude that $P(y;\sigma_{X_1},\sigma_{X_2})$ is not identifiable.

On the other hand, if $P(x_2|x_1,z;\sigma_{X_2})$ is independent of Z, say $P(x_2|x_1,z;\sigma_{X_2}) = P^*(x_2|x_1)$ (or $\sigma_{X_2} = do(x_2)$), then the set Z_D of variables in Z that are ancestors of Y in G_{σ_X} becomes empty (see Figure 2(b)), and, by Theorem 1, the identifiability of $P(y;\sigma_{X_1},\sigma_{X_2})$ depends on the identifiability of $Q[\{Y\}]$. In fact, in this case, Eq. (28) becomes

$$\begin{aligned} &P(y;\sigma_{X_1},\sigma_{X_2}=d_{X_1}) \\ &= \sum_{x_1,x_2} P(x_1;\sigma_{X_1})P^*(x_2|x_1)\sum_z Q[\{Y,Z\}] \\ &= \sum_{x_1,x_2} P(x_1;\sigma_{X_1})P^*(x_2|x_1)Q[\{Y\}] \end{aligned} \quad (29)$$

From Eq. (27) we obtain

$$\begin{aligned} Q[\{X_1,Z,Y\}] &= P(v)/P(x_2|x_1,z) \\ &= P(y|x_1,x_2,z)P(x_1,z). \end{aligned} \quad (30)$$

It can be shown (or confirmed) that

$$\begin{aligned} Q[\{Y\}] &= \frac{\sum_z Q[\{X_1,Z,Y\}]}{\sum_{y,z} Q[\{X_1,Z,Y\}]} \\ &= \sum_z P(y|x_1,x_2,z)P(z|x_1). \end{aligned} \quad (31)$$

We obtain

$$P(y;\sigma_{X_1},\sigma_{X_2}=d_{X_1}) = \sum_{x_1,x_2} P(x_1;\sigma_{X_1})P^*(x_2|x_1)\sum_z P(y|x_1,x_2,z)P(z|x_1). \quad (32)$$

And in particular, the unconditional plan is identified as

$$P_{x_1,x_2}(y) = Q[\{Y\}] = \sum_z P(y|x_1,x_2,z)P(z|x_1). \quad (33)$$

5.2 Identification via conditional causal effects?

Pearl (2000) has suggested that dynamic sequential plans involving conditional and random interventions

may be identified by identifying conditional causal effects of the form $P_x(y|z)$. For interventions on a single variable X, we can show [Pearl, 2000, Section 4.2] that

$$P(y; \sigma_X = do(g(z))) = \sum_z P_x(y|z)|_{x=g(z)} P(z),$$

and

$$P(y; \sigma_X = d_Z) = \sum_{x,z} P_x(y|z) P^*(x|z) P(z).$$

Therefore it appears that $P(y; \sigma_X)$ is identifiable if and only if $P_x(y|z)$ is identifiable.

This idea was generalized to dynamic sequential plans. Consider a plan involving a sequence of conditional interventions $\sigma_{X_i} = do(g_i(C_i))$. Let $Z_{\sigma_X} = Z \cap (\cup_i C_i)$ be the set of conditioning variables in the strategy σ_X. Pearl (2006) shows that

$$P(y; \sigma_X) = \sum_{z_{\sigma_X}} P_{x_z}(y|z_{\sigma_X}) P_{x_z}(z_{\sigma_X}), \qquad (34)$$

where x_z are the values attained by X when the conditioning set Z_{σ_X} takes the values z_{σ_X}. Pearl then suggests that sequential conditional plans can be identified by identifying conditional causal effects $P_x(y|z)$ and $P_x(z)$. This motivated the study of the identifiability of conditional causal effects and complete algorithms have been developed in [Tian, 2004, Shpitser and Pearl, 2006a].

Next we show that although the identification of $P_{x_z}(y|z_{\sigma_X})$ and $P_{x_z}(z_{\sigma_X})$ is sufficient for identifying $P(y; \sigma_X)$, it is nonetheless not necessary. Rewrite Eq. (34) in the following

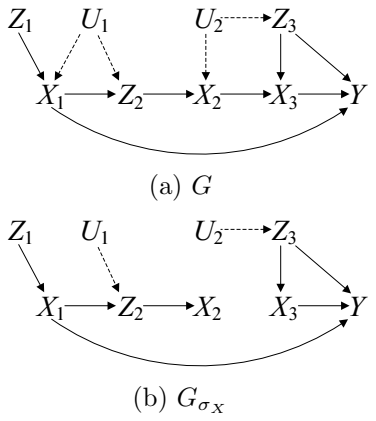
$$P(y; \sigma_X) = \sum_{x, z_{\sigma_X}} \delta(x_i, g_i(C_i)) P_x(y, z_{\sigma_X}) \qquad (35)$$

$$= \sum_{x, z_{\sigma_X}} \delta(x_i, g_i(C_i)) \sum_{z \setminus z_{\sigma_X}} Q[Y, Z] \qquad (36)$$

Comparing the reduction from Eq. (14) into (17) with the reduction from (14) to (36), we obtain that if $X_D = X$ then (36) is equivalent to (17), otherwise Eq. (36) may be further reduced in that more factors could be summed out from $Q[Y, Z]$. We obtain the following conclusion

- If all the variables in X are ancestors of Y in G_{σ_X}, then the sequential plan $P(y; \sigma_X)$ can be identified by identifying the causal effects $P_x(y, z_{\sigma_X})$, otherwise it is possible that $P(y; \sigma_X)$ is identifiable even if $P_x(y, z_{\sigma_X})$ is not.

We demonstrate this point with an example. Consider the problem of identifying $P(y; \sigma_X)$ where



(a) G

(b) G_{σ_X}

Figure 3: An example causal graph

$\sigma_X = \{\sigma_{X_1} = do(g_1(Z_1)), \sigma_{X_2} = do(g_2(Z_2)), \sigma_{X_3} = do(g_3(Z_3))\}$ in Figure 3(a). The graph G_{σ_X} is shown in Figure 3(b). We have $Z_D = \{Z_1, Z_3\}$, and Theorem 1 calls for identifying $Q[\{Y, Z_1, Z_3\}]$ which can be shown to be identifiable. On the other hand, $P_{x_1 x_2 x_3}(y, z_1, z_2, z_3) = Q[\{Y, Z_1, Z_2, Z_3\}]$ is not identifiable. More specifically, given the observational distribution

$$P(v) = P(y|x_1, x_3, z_3) P(x_3|x_2, z_3) P(z_1) Q[\{X_2, Z_3\}] \; Q[\{X_1, Z_2\}], \qquad (37)$$

we want to identify

$$P(y; \sigma_X) = \sum_{x_i, z_i} \prod_i \delta(x_i, g_i(z_i)) P(y|x_1, x_3, z_3) P(z_1) \; Q[\{Z_3\}] Q[\{Z_2\}] \qquad (38)$$

$$= \sum_{x_1, x_3, z_1, z_3} \delta(x_1, g_1(z_1)) \delta(x_3, g_3(z_3)) P(y|x_1, x_3, z_3) \; P(z_1) Q[\{Z_3\}], \qquad (39)$$

where $Q[\{Z_3\}]$ can be identified as

$$Q[\{Z_3\}] = \sum_{u_2} P(z_3|u_2) P(u_2) = P(z_3). \qquad (40)$$

We obtain

$$P(y; \sigma_X) = \sum_{z_1, z_3} P(y|g_1(z_1), g_3(z_3), z_3) P(z_1) P(z_3). \qquad (41)$$

On the other hand,

$$P_{x_1 x_2 x_3}(y, z_1, z_2, z_3) = P(y|x_1, x_3, z_3) P(z_1) Q[\{Z_3\}] Q[\{Z_2\}] = P(y|x_1, x_3, z_3) P(z_1) P(z_3) Q[\{Z_2\}] \qquad (42)$$

is not identifiable since $Q[\{Z_2\}]$ is not identifiable. In fact the conditional causal effect

$$P_{x_1x_2x_3}(y|z_1, z_2, z_3) = P(y|x_1, x_3, z_3) \quad (43)$$

is identifiable but the causal effect

$$P_{x_1x_2x_3}(z_1, z_2, z_3) = P(z_1)P(z_3)Q[\{Z_2\}] \quad (44)$$

is not identifiable.

6 Conclusion

We present a method for identifying dynamic sequential plans. A closed-form expression for the probability of the outcome variables under a dynamic plan can be obtained in terms of the observed distribution, by using the algorithms for identifying causal effects available in the literature.

Acknowledgments

This research was partly supported by NSF grant IIS-0347846.

References

[Dawid and Didelez, 2005] A.P. Dawid and V. Didelez. Identifying the consequences of dynamic treatment strategies. Technical report, Department of Statistical Science, University College London, UK, 2005.

[Dawid, 2002] A.P. Dawid. Influence diagrams for causal modelling and inference. *International Statistical Review*, 70(2), 2002.

[Didelez *et al.*, 2006] V. Didelez, A.P. Dawid, and S. Geneletti. Direct and indirect effects of sequential treatments. In *Proceedings of the 22nd Annual Conference on Uncertainty in Artifical Intelligence*, pages 138–146, 2006.

[Heckerman and Shachter, 1995] D. Heckerman and R. Shachter. Decision-theoretic foundations for causal reasoning. *Journal of Artificial Intelligence Research*, 3:405–430, 1995.

[Huang and Valtorta, 2006] Y. Huang and M. Valtorta. Identifiability in causal bayesian networks: A sound and complete algorithm. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 1149–1154, Menlo Park, CA, July 2006. AAAI Press.

[Lauritzen, 2000] S. Lauritzen. Graphical models for causal inference. In O.E. Barndorff-Nielsen, D. Cox, and C. Kluppelberg, editors, *Complex Stochastic Systems*, chapter 2, pages 67–112. Chapman and Hall/CRC Press, London/Boca Raton, 2000.

[Pearl and Robins, 1995] J. Pearl and J.M. Robins. Probabilistic evaluation of sequential plans from causal models with hidden variables. In P. Besnard and S. Hanks, editors, *Uncertainty in Artificial Intelligence 11*, pages 444–453. Morgan Kaufmann, San Francisco, 1995.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, CA, 1988.

[Pearl, 2000] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, NY, 2000.

[Pearl, 2006] J. Pearl. Comment on identifying conditional plans. 2006. http://www.mii.ucla.edu/causality/?p=36#comments.

[Robins, 1986] J.M. Robins. A new approach to causal inference in mortality studies with a sustained exposure period – applications to control of the healthy workers survivor effect. *Mathematical Modeling*, 7:1393–1512, 1986.

[Robins, 1987] J.M. Robins. A graphical approach to the identification and estimation of causal parameters in mortality studies with sustained exposure periods. *Journal of Chronic Diseases*, 40(Suppl 2):139S–161S, 1987.

[Shpitser and Pearl, 2006a] I. Shpitser and J. Pearl. Identification of conditional interventional distributions. In R. Dechter and T.S. Richardson, editors, *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 437–444. AUAI Press, July 2006.

[Shpitser and Pearl, 2006b] I. Shpitser and J. Pearl. Identification of joint interventional distributions in recursive semi-markovian causal models. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 1219–1226, Menlo Park, CA, July 2006. AAAI Press.

[Spirtes *et al.*, 1993] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. Springer-Verlag, New York, 1993.

[Tian and Pearl, 2003] J. Tian and J. Pearl. On the identification of causal effects. Technical Report R-290-L, Department of Computer Science, University of California, Los Angeles, 2003. http://www.cs.iastate.edu/ ∼jtian/r290-L.pdf.

[Tian, 2004] J. Tian. Identifying conditional causal effects. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2004.

Hierarchical POMDP Controller Optimization by Likelihood Maximization

Marc Toussaint
Computer Science
TU Berlin
Berlin, Germany
mtoussai@cs.tu-berlin.de

Laurent Charlin
Computer Science
University of Toronto
Toronto, Ontario, Canada
lcharlin@cs.toronto.edu

Pascal Poupart
Computer Science
University of Waterloo
Waterloo, Ontario, Canada
ppoupart@cs.uwaterloo.ca

Abstract

Planning can often be simplified by decomposing the task into smaller tasks arranged hierarchically. Charlin et al. [4] recently showed that the hierarchy discovery problem can be framed as a non-convex optimization problem. However, the inherent computational difficulty of solving such an optimization problem makes it hard to scale to real-world problems. In another line of research, Toussaint et al. [18] developed a method to solve planning problems by maximum-likelihood estimation. In this paper, we show how the hierarchy discovery problem in partially observable domains can be tackled using a similar maximum likelihood approach. Our technique first transforms the problem into a dynamic Bayesian network through which a hierarchical structure can naturally be discovered while optimizing the policy. Experimental results demonstrate that this approach scales better than previous techniques based on non-convex optimization.

1 Introduction

Planning in partially observable domains is notoriously difficult. However, many planning tasks naturally decompose into subtasks that may be arranged hierarchically. For instance, the design of a soccer playing robot is often decomposed into low-level skills such as intercepting the ball, controlling the ball, passing the ball, etc. [16]. Similarly, prompting systems that assist older adults with activities of daily living (e.g., handwashing [8]) can be naturally decomposed into subtasks for each step of an activity. When a decomposition or hierarchy is known a priori, several approaches have demonstrated that planning can be simplified and performed faster [13, 7]. However, the hierarchy is not always known or easy to specify, and the optimal policy may only decompose *approximately*. To that effect, Charlin et al. [4] showed how a hierarchy can be discovered automatically by formulating the planning problem as a non-convex quartically constrained optimization problem with variables corresponding to the parameters of the policy, including its hierarchical structure. Unfortunately, the inherent computational difficulty of solving this optimization problem prevents the approach from scaling to real-world problems. Furthermore, it is not clear that automated hierarchy discovery simplifies planning since the space of policies remains the same.

We propose an alternative approach that demonstrates that hierarchy discovery (i) can be done efficiently and (ii) performs a policy search with a different bias than non-hierarchical approaches that is advantageous when there exists good hierarchical policies. The approach combines Murphy and Paskin's [10] factored encoding of hierarchical structures (see also [17]) into a dynamic Bayesian network (DBN) with Toussaint et al.'s [18] maximum-likelihood estimation technique for policy optimization. More precisely, we encode POMDPs with hierarchical controllers into a DBN in such a way that the policy and hierarchy parameters are entries of some conditional probability tables. We also consider factored policies that are more general than hierarchical controllers. The policy and hierarchy parameters are optimized with the expectation-maximization (EM) algorithm [5]. Since each iteration of EM essentially consists of inference queries, the approach scales easily.

Sect. 2 briefly introduces partially observable Markov decision processes, controllers and policy optimization by maximum likelihood estimation. Sect. 3 reviews previous work on hierarchical modeling and how to use a dynamic Bayesian network to encode a hierarchical structure. Sect. 4 describes our proposed approach, which combines a dynamic Bayesian network encoding with maximum likelihood estimation to simultane-

ously optimize a hierarchy and the controller. Sect. 5 demonstrates the scalability of the proposed approach on benchmark problems. Finally, Sect. 6 summarizes the paper and discusses future work.

2 Background

Throughout the paper we denote random variables by upper case letters (e.g., X), values of random variables by their corresponding lower case letters (e.g., $x \in dom(X)$) and sets of values by upper case letters with math calligraphy (e.g., $\mathcal{X} = \{x_1, x_2, x_3\}$). We now review POMDPs (Sect. 2.1), how to represent policies as finite state controllers (Sect. 2.2) and how to optimize bounded controllers (Sect. 2.3).

2.1 POMDPs

Partially observable Markov decision processes (POMDPs) provide a natural and principled framework for planning. A POMDP can be formally defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, p_s, p_{s'|as}, p_{o'|s'a}, r_{as} \rangle$ where $\mathcal{S}$ is the set of states s, $\mathcal{A}$ is the set of actions a, $\mathcal{O}$ is the set of observations o, $p_s = \Pr(S_0 = s)$ is the initial state distribution (a.k.a. initial belief), $p_{s'|as} = \Pr(S_{t+1} = s' \,|\, A_t = a, S_t = s)$ is the transition distribution, $p_{o'|s'a} = \Pr(O_{t+1} = o' \,|\, S_{t+1} = s', A_t = a)$ is the observation distribution and $r_{as} = R(A_t = a, S_t = s)$ is the reward function. Throughout the paper, it is assumed that $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{O}$ are finite and discrete. The goal is to select actions to maximize the rewards. At any point in time, the information available to select the next action consists of the history of past actions and observations. Hence a policy π is defined as a mapping from histories to actions. However, since histories grow with time, it is common practice to summarize histories with a fixed-length sufficient statistic such as the belief distribution $b_s = \Pr(S = s)$, which corresponds to the state distribution (conditioned on the history of past actions and observations). The belief distribution b can be updated at each time step, based on the action a taken and the observation o' made according to Bayes' theorem: $b^{ao'}_{s'} = k \sum_s b_s p_{s'|as} p_{o'|s'a}$ (k is a normalization constant). Policies can then be defined as mappings from beliefs to actions (e.g., $\pi(b) = a$). The value $V^\pi(b)$ of a policy π starting in belief b is measured by the discounted sum of expected rewards: $V^\pi(b) = \sum_t \gamma^t E_{b_t|\pi}[r_{\pi(b_t)b_t}]$ where $r_{ab} = \sum_s b_s r_{as}$. An optimal policy π^* is a policy with the highest value V^* for all beliefs: $V^*(b) \geq V^\pi \; \forall \pi, b$. The optimal value function also satisfies Bellman's equation: $V^*(b) = \max_a r_{ab} + \sum_{o'} p_{o'|ab} V^*(b^{ao'})$ where $p_{o'|ab} = \sum_{ss'} b_s p_{s'|as} p_{o'|s'a}$.

2.2 Finite State Controllers

A convenient representation for an important class of policies consists of finite state controllers [6]. Instead of using beliefs as sufficient statistics of histories, the idea is to use a finite internal memory to retain relevant bits of information from histories. Each configuration of this memory can be thought of as a node in a finite state controller, where nodes select actions to be executed and edges indicate how to update nodes based on the observations received. A controller with a finite set $\mathcal{N}$ of nodes n can encode a stochastic policy π with three distributions: $\Pr(N_0 = n) = p_n$ (initial node distribution), $\Pr(A_t = a \,|\, N_t = n) = p_{a|n}$ (action selection distribution) and $\Pr(N_{t+1} = n' \,|\, N_t = n, O_{t+1} = o') = p_{n'|no'}$ (successor node distribution). Such a policy can be executed by starting in a node n sampled from p_n, executing an action a sampled from $p_{a|n}$, receiving observation o', transitioning to node n' sampled from $p_{n'|no'}$ and so on. The value of a controller can be computed by solving a linear system: $V_{ns} = \sum_a p_{a|n} [r_{as} + \gamma \sum_{s'o'n'} p_{s'|as} p_{o'|s'a} p_{n'|no'} V_{n's'}] \; \forall ns$. The value at a given belief b is then $V^\pi(b) = \sum_n \sum_s b_s p_n V_{ns}$.

2.3 Policy Optimization

Several techniques have been proposed to optimize controllers of a given size, including gradient ascent [9], stochastic local search [2], bounded policy iteration [14], non-convex quadratically constrained optimization [1] and likelihood maximization [18]. We briefly describe the last technique since we will use it in Sect. 4.

Toussaint et al. [18] recently proposed to convert POMDPs into equivalent dynamic Bayesian networks (DBNs) by normalizing the rewards and to optimize a policy by maximizing the likelihood of the normalized rewards. Let $\tilde{R}$ be a binary variable corresponding to normalized rewards. The reward function r_{as} is then replaced by a reward distribution $p_{\tilde{r}|sat} = \Pr(\tilde{R} = \tilde{r} \,|\, A_t = a, S_t = s, T = t)$ that assigns probability $r_{as}/(r_{max} - r_{min})$ to $\tilde{R} = 1$ and $1 - r_{as}/(r_{max} - r_{min})$ to $\tilde{R} = 0$ ($r_{min} = \min_{as} r_{as}$ and $r_{max} = \max_{as} r_{as}$). An additional time variable T is introduced to simulate the discount factor and the summation of rewards. Since a reward is normally discounted by a factor γ^t when earned t time steps in the future, the prior $p_t = \Pr(T = t)$ is set to $\gamma^t(1-\gamma)$ where the factor $(1-\gamma)$ ensures that $\sum_{t=0}^{\infty} p_t = 1$. The resulting dynamic Bayesian network is illustrated in Fig. 1. It can be thought of as a mixture of finite processes of length t with a 0-1 reward $\tilde{R}$ earned at the end of the process. The nodes N_t encode the internal memory of the controller. Given the controller distributions p_n, $p_{a|n}$ and $p_{n'|no'}$, it is possible to evaluate the controller

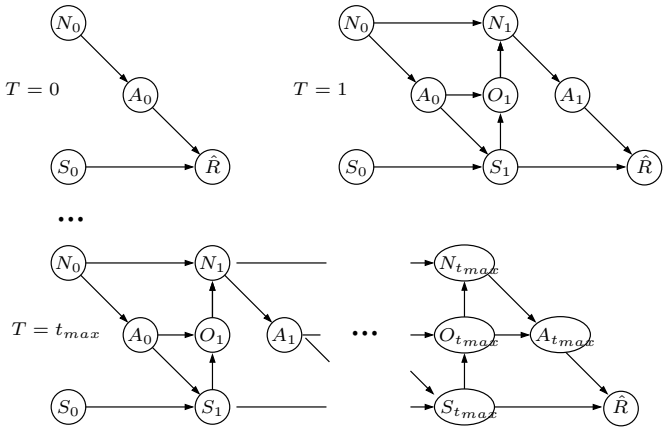


Figure 1: POMDP represented as a mixture of finite DBNs. For an infinite horizon, a large enough t_{max} can be selected at runtime to ensure that the approximation error is small.

by computing the likelihood of $\tilde{R} = 1$. More precisely, $V^{\pi}(p_s) = (\Pr(\tilde{R}{=}1) - r_{min})/[(r_{max} - r_{min})(1-\gamma)]$.

Optimizing the policy can be framed as maximizing the likelihood of $\tilde{R} = 1$ by varying the distributions p_n, $p_{a|n}$ and $p_{n'|no'}$ encoding the policy. Toussaint et al. use the expectation-maximization (EM) algorithm. Since EM is guaranteed to increase the likelihood at each iteration, the controller's value increases monotonically. However, EM is not guaranteed to converge to a global optimum. An important advantage of the EM algorithm is its simplicity both conceptually and computationally. In particular, the computation consists of inference queries that can be computed using a variety of exact and approximate algorithms.

3 Hierarchical Modeling

While optimizing a bounded controller allows an effective search in the space of bounded policies, such an approach is clearly suboptimal since the optimal controller of many problems grows doubly exponentially with the planning horizon and may be infinite for infinite horizons. Alternatively, hierarchical representations permit the representation of structured policies with exponentially fewer parameters. Several approaches were recently explored to model and learn hierarchical structures in POMDPs. Pineau et al. [13] sped up planning by exploiting a user specified action hierarchy. Hansen et al. [7] proposed hierarchical controllers and an alternative planning technique that also exploits a user specified hierarchy. Charlin et al. [4] proposed recursive controllers (which subsume hierarchical controllers) and an approach that discovers the hierarchy while optimizing a controller. We briefly review recursive controllers in Sect. 3.1 since we will empirically compare our approach to the non-convex optimization techniques used to optimize recursive controllers. In another line of research, Murphy and Paskin [10] proposed to model hierarchical hidden Markov models (HMMs) with a dynamic Bayesian network (DBN). Theocharous et al. [17] also used DBNs to model hierarchical POMDPs. We briefly review this DBN encoding in Sect. 3.2 since we will use it in our approach to model factored controllers.

3.1 Recursive Controllers

A recursive controller [4] consists of a recursive automaton with concrete nodes n and abstract nodes $\bar{n}$. Abstract nodes call a subcontroller before selecting an action. A controller is said to be recursive when it can call itself, essentially encoding an infinite hierarchy. Formally, a recursive controller is parametrized by an action selection distribution for each node (e.g., $p_{a|n}$ and $p_{a|\bar{n}}$), a successor node distributions for each node (e.g., $p_{n'|no'}$ and $p_{n'|\bar{n}o'}$) and a child node distribution for each abstract node (e.g., $p_{n'|\bar{n}}$)[1]. Execution of a recursive controller is performed by executing the action selected by each node visited and continuing to the successor node selected by the observation made. However, when an abstract node is visited, before executing the action selected, its subcontroller is called and started in the child node selected by the child node distribution. A subcontroller returns control to its parent node when a special end node is reached. Charlin et al. [4] show that optimizing a recursive controller with a fixed number of concrete and abstract nodes can be framed as a non-convex quartically constrained optimization problem. The hierarchical structure is discovered as the controller is optimized since the variables of the optimization problem include the child node distributions which implicitly encode the hierarchy. Three techniques based on a general non-linear solver, a mixed-integer non-linear approximation and a form of bounded hierarchical policy iteration are experimented with, but do not scale beyond toy problems. Furthermore, Charlin et al. do not demonstrate whether searching in the space of hierarchical controllers can speed up planning. Although it is clear that planning is simplified when a hierarchy is given a priori since the policy space is reduced, it is not clear that hierarchy discovery is beneficial since the policy space remains the same while the parameter space changes. In Sect. 5, we demonstrate that hierarchy discovery can be beneficial when a simple hierarchical policy of high value exists.

[1] The $p_{a|n}$ and $p_{n'|no'}$ distributions are combined in one distribution $p_{n'a|no'}$ in [14]

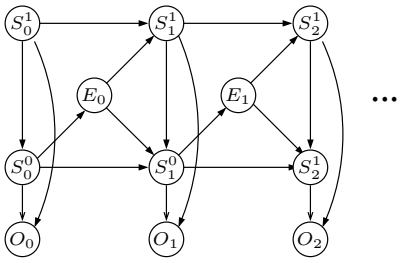


Figure 2: DBN encoding of a 2-level hierarchical HMM.

3.2 Hierarchical HMMs

Murphy and Paskin [10] proposed to model hierarchical hidden Markov models (HMMs) as dynamic Bayesian networks (DBNs). The idea is to convert a hierarchical HMM of L levels into a dynamic Bayesian network of L state variables, where each variable encodes abstract states at the corresponding level. Here, abstract states can only call sub-HMMs at the previous level. Fig. 2 illustrates a two-level hierarchical HMMs encoded as a DBN. The state variables S_t^l are indexed by the time step t and the level l. The E_t variables indicate when a base-level sub-HMM has ended, returning its control to the top level HMM. The top-level abstract state transitions according to the top HMM, but only when the exit variable E_t indicates that the base-level concrete state is an exit state. The base-level concrete state transitions according to the base-level HMM. When an exit state is reached, the next base-level state is determined by the next top-level abstract state. Factored HMMs subsume hierarchical HMMs in the sense that there exists an equivalent factored HMM for every hierarchical HMM. In Sect. 4.1, we will use a similar technique to convert hierarchical controllers into factored controllers.

4 Factored Controllers

We propose to combine the DBN encoding techniques of Murphy et al. [10] and Toussaint et al. [18] to convert a POMDP with a hierarchical controller into a mixture of DBNs. The hierarchy and the controller are simultaneously optimized by maximizing the reward likelihood of the DBN. We also consider factored controllers which subsume hierarchical controllers.

4.1 DBN Encoding

Fig. 3a illustrates two consecutive slices of one DBN in the mixture (rewards are omitted) for a three-level hierarchical controller. Consider a POMDP defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, p_s, p_{s'|as}, p_{o'|s'a}, r_{as} \rangle$ and a three-level hierarchical (non-recursive) controller defined by the tuple $\langle p_{a|n^l}, p_{n^{l-1}|n^l}, p_{n'^l|n^l o'} \rangle \ \forall l$. The conditional probability distributions of the mixture of DBNs (denoted by $\hat{p}$) are:

- transition distribution: $\hat{p}_{s'|as} = p_{s'|as}$
- observation distribution: $\hat{p}_{o'|s'a} = p_{o'|s'a}$
- reward distribution: $\hat{p}_{\tilde{r}|as} = (r_{as} - r_{min})/(r_{max} - r_{min})$
- mixture distribution: $\hat{p}_t = (1-\gamma)\gamma^t$
- action distribution: $\hat{p}_{a|n^0} = p_{a|n^0}$
- base level node distribution: $\hat{p}_{n'^0|n^0 n'^1 o' e^0}$
  $= \begin{cases} p_{n'^0|n'^1} & \text{if } e^0 = exit \\ p_{n'^0|o' n^0} & \text{otherwise} \end{cases}$
- middle level node distribution: $\hat{p}_{n'^1|n^1 n'^2 o' e^0 e^1}$
  $= \begin{cases} p_{n'^1|n'^2} & \text{if } e^1 = exit \\ p_{n'^1|o' n^1} & \text{if } e^0 = exit \text{ and } e^1 \neq exit \\ \delta_{n'^1 n^1} & \text{otherwise} \end{cases}$
- top level node distribution: $\hat{p}_{n'^2|o' n^2 e^1}$
  $= \begin{cases} p_{n'^2|o' n^2} & \text{if } e^1 = exit \\ \delta_{n'^2 n^2} & \text{otherwise} \end{cases}$
- base-level exit distribution: $\hat{p}_{e^0|n^0}$
  $= \begin{cases} 1 & \text{if } n^0 \text{ is an end node} \\ 0 & \text{otherwise} \end{cases}$
- middle-level exit distribution: $\hat{p}_{e^1|n^1 e^0}$
  $= \begin{cases} 1 & \text{if } e^0 = exit \text{ and } n^1 \text{ is an end node} \\ 0 & \text{otherwise} \end{cases}$

While the E_t^l variables help clarify when the end of a sub-controller is reached, they are not necessary. Eliminating them yields a simpler DBN illustrated in Fig. 3b. The conditional probability distributions of the N_t^l variables become:

- base level node distribution: $\hat{p}_{n'^0|n^0 n'^1 o'}$
  $= \begin{cases} p_{n'^0|n'^1} & \text{if } n^0 \text{ is an end node} \\ p_{n'^0|o' n^0} & \text{otherwise} \end{cases}$
- middle level node distribution: $\hat{p}_{n'^1|n^1 n'^2 o'}$
  $= \begin{cases} p_{n'^1|n'^2} & \text{if } n^1 \text{ and } n^0 \text{ are end nodes} \\ p_{n'^1|o' n^1} & \text{if } n^0 \text{ is an end node, but not } n^1 \\ \delta_{n'^1 n^1} & \text{otherwise} \end{cases}$
- top level node distribution: $\hat{p}_{n'^2|n^2 o' e^1}$
  $= \begin{cases} p_{n'^2|n^2 o'} & \text{if } n^1 \text{ and } n^0 \text{ are end nodes} \\ \delta_{n'^2 n^2} & \text{otherwise} \end{cases}$

Note that ignoring the above constraints in the conditional distributions yields a *factored controller* that is more flexible than a hierarchical controller since the conditional probability distributions of the N_t^l variables do not have to follow the structure imposed by a hierarchy

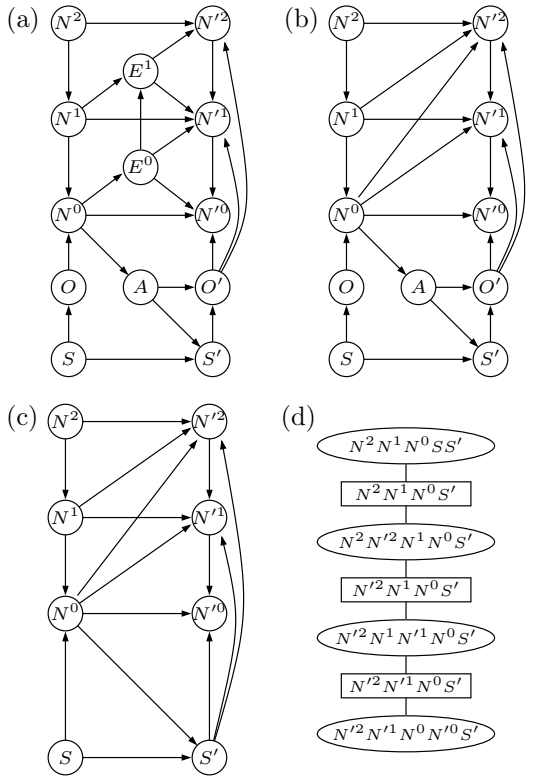


Figure 3: (a) Two slices of the DBN encoding the hierarchical POMDP controller. (b) A version where exit variables are eliminated. (c) Variables O and A are eliminated. (d) The corresponding junction tree (or rather chain) for inference.

4.2 Maximum Likelihood Estimation

Following Toussaint et al.'s technique [18], we optimize a factored controller by maximizing the reward likelihood. Since the policy parameters are conditional probability distributions of the DBN, the EM algorithm can be used to optimize them. Computation alternates between the E and M steps below. We denote by n^{top} and n^{base} the top and base nodes in a given time slice. We also denote by $\phi(V)$ and $\phi(v)$ the parents of V and a configuration of the parents of V.

E-step: expected frequency of the hidden variables
$E_{n^{top}} = \Pr(N_0^{top}=n^{top}|\tilde{R}=1)$
$E_{an^{base}} = \sum_t \Pr(A_t=a, N_t^{base}=n^{base}|\tilde{R}=1)$
$E_{n'^l\phi(n'^l)} =$
$\sum_t \Pr(N_{t+1}^l=n'^l, \phi(N_{t+1}^l)=\phi(n_{t+1}^l)|\tilde{R}=1)\ \forall l$

M-step: relative frequency computation
$p_{n^{top}} = E_{n^{top}} / \sum_{n^{top}} E_{n^{top}}$
$p_{a|n^{base}} = E_{an^{base}} / \sum_a E_{an^{base}}$
$p_{n'^l|\phi(n'^l)} = E_{n'^l\phi(n'^l)} / \sum_{n'^l} E_{n'^l\phi(n'^l)}\ \forall l$

4.2.1 Parameter initialization

W.l.o.g. we initialize the start node N_0^{top} of the top layer to be the first node (i.e., $\Pr(N_0^{top}=1)=1$). The node conditional distributions $p_{n'^l|\phi(n'^l)}$ are initialized randomly as a mixture of three distributions:

$$p_{n'^l|\phi(n'^l)} \propto c_1 + c_2\mathcal{U}_{n'^l\phi(n'^l)} + c_3\delta_{n'^ln^l}$$

The mixture components are a uniform distribution, a random distribution $\mathcal{U}_{\phi(n'^l)}$ (an array of uniform random numbers in $[0,1]$), and a term enforcing n^l to stay unchanged. For the node distributions at the base level we choose $c_1=1, c_2=1, c_3=0$ and for all other levels we choose $c_1=1, c_2=1, c_3=10$. Similarly we initialize the action probabilities as

$$p_{a|n^{base}} \propto c_1 + c_2\mathcal{U}_{an^{base}} + c_3\delta_{a(n^{base}\%a)}$$

with $c_1=1, c_2=1, c_3=100$, where the last term enforces each node $n^{base}=i$ to be associated with action $a=i\%a$.

4.2.2 E-step

To speed up the computation of the inference queries in the E-step, we compute intermediate terms using a forward-backward procedure. Let t_{max} be the largest value of T, then a simple scheme that answers each query separately takes $O(t_{max}^2)$ time since there are $O(t_{max})$ queries and each query takes $O(t_{max})$ time to run over the entire network. However, since part of the computation is duplicated in several queries, it is possible to compute intermediate terms α and β in $O(t_{max})$ time from which each expectation can be computed in constant time (w.r.t. t_{max}). To simplify notation, $\mathbf{N}$ and $\mathbf{n}$ denote all the nodes and their joint configuration in a given time slice.

Forward term: $\alpha_{\mathbf{n}s}^t = \Pr(\mathbf{N}_t=\mathbf{n}, S_t=s)$
$\alpha_{\mathbf{n}s}^0 = p_{\mathbf{n}}p_s$
$\alpha_{\mathbf{n}'s'}^t = \sum_{\mathbf{n},s} \alpha_{\mathbf{n}s}^{t-1} p_{\mathbf{n}'s'|\mathbf{n}s}$

Backward term: $\beta_{\mathbf{n}s}^\tau = \Pr(\tilde{R}=1|\mathbf{N}_{t-\tau}=\mathbf{n}, S_{t-\tau}=s, T=t)$
$\beta_{\mathbf{n}s}^0 = \sum_a p_{a|\mathbf{n}} r_{as}$
$\beta_{\mathbf{n}s}^\tau = \sum_{\mathbf{n}',s'} p_{\mathbf{n}'s'|\mathbf{n}s}\beta_{\mathbf{n}'s'}^{\tau-1}$

To fully take advantage of the structure of the DBN, we first marginalize the DBN w.r.t. the observations and actions to get the DBN in Fig. 3c. This 2-slice DBN corresponds to the joint transition distribution $p_{\mathbf{n}'s'|\mathbf{n}s}$ used in the above equations. Then we compile this 2-slice DBN into the junction tree (actually junction chain) given in Fig. 3d.

Let $\beta_{\mathbf{n}s} = \sum_\tau \Pr(T=\tau)\beta_{\mathbf{n}s}^\tau$ and $\alpha_{\mathbf{n}s} = \sum_t \Pr(T=t)\alpha_{\mathbf{n}s}^t$, then the last two expectations of the E-step

can be computed as follows:[2]

$$E_{an^{base}} \propto \sum_{s,\mathbf{n}-\{n^{base}\}} \alpha_{\mathbf{n}s} p_{a|n^{base}} \big[r_{as} + \sum_{s',o',\mathbf{n}'} p_{s'|as} p_{o'|s'a} p_{\mathbf{n}'|o'\mathbf{n}} \beta_{\mathbf{n}'s'}\big]$$
$$E_{n'^l \phi(n'^l)} \propto \sum_{s,a,\mathbf{n}-\phi(n'^l)} \alpha_{\mathbf{n}s} p_{a|n^{base}} \big[r_{as} + \sum_{s',n'^{-l}} p_{s'|as} p_{o'|s'a} p_{\mathbf{n}'|o'\mathbf{n}} \beta_{\mathbf{n}'s'}\big] \ \forall l$$

4.2.3 M-step

The standard M-step adjusts each parameter $p_{v|\phi(v)}$ by normalizing the expectations computed in the E-step, i.e., $p^{\text{new}}_{v|\phi(v)} \propto E_{v\phi(v)}$. To speed up convergence, we instead use a variant that performs a soften *greedy M-step*. In the greedy M-step, each parameter $p^{\text{new}}_{v|\phi(v)}$ is greedily set to 1 when $v = \text{argmax}_{\bar{v}} f_{\bar{v}\phi(\bar{v})}$ and 0 otherwise, where $f_{v\phi(v)} = E_{v\phi(v)}/p^{\text{old}}_{v|\phi(v)}$. The greedy M-step can be thought of as the limit of an infinite sequence of alternating partial E-step and standard M-step where the partial E-step keeps f fixed. The combination of a standard M-step with this specific partial E-step updates $p_{v|\phi(v)}$ by a multiplicative factor proportional to $f_{v\phi(v)}$. In the limit, the largest $f_{v\phi(v)}$ ends up giving all the probability to the corresponding $p_{v|\phi(v)}$. EM variants with certain types of partial E-steps ensure monotonic improvement of the likelihood when the hidden variables are independent [11]. This is not the case here, however by softening the greedy M-step we can still obtain monotonic improvement most of the time while speeding up convergence. We update $p_{v|\phi(v)}$ as follows:

$$v^* = \operatorname*{argmax}_v f_{v\phi(v)}$$
$$p^{\text{new}}_{v|\phi(v)} \propto p^{\text{old}}_{v|\phi(v)} [\delta_{vv^*} + c + \epsilon] \ .$$

For $c = 0$ and $\epsilon = 0$ this is the greedy M-step. We use $c = 3$ which softens (shortens) the step and improves convergence. Furthermore, adding small Gaussian noise $\epsilon \sim \mathcal{N}(0, 10^{-3})$ helps to escape local minima.

4.2.4 Complexity

For a flat controller, the number of parameters (neglecting normalization) is $|\mathcal{O}||\mathcal{N}|^2$ for $p_{n'|o'n}$ and $|\mathcal{A}||\mathcal{N}|$ for $p_{a|n}$. The complexity of the forward (backward) procedure is $O(t_{max}(|\mathcal{N}||\mathcal{S}|^2 + |\mathcal{N}|^2|\mathcal{S}|))$ where the two terms correspond to the size of the two cliques for inference in the 2-slice DBN after O and A are eliminated. The complexity of computing the expectations from α and β is $O(|\mathcal{N}||\mathcal{A}|(|\mathcal{S}|^2 + |\mathcal{S}||\mathcal{O}|) + |\mathcal{N}|^2|\mathcal{S}||\mathcal{O}|)$, which corresponds to the clique sizes of the 2-slice DBN including O and A.

In comparison, 2-level hierarchical and factored controllers with $|\mathcal{N}^{top}| = |\mathcal{N}^{base}| = |\mathcal{N}|^{0.5}$ nodes at each level have fewer parameters and a smaller complexity, but also a smaller policy space due to the structure imposed by the hierarchy/factorization. While there is a tradeoff between policy space and complexity, hierarchical and factored controllers are often advantageous in practice since they can find more quickly a good hierarchical/factored policy when there exists one.

[2]The first expectation of the E-step does not need to be computed since $\Pr(N_0^{top} = 1) = 1$.

Table 1: Number of parameters and computational complexity for the flat controller with $|\mathcal{N}|$ nodes and a 2-layer factored controller with $|\mathcal{N}^{top}| = |\mathcal{N}^{base}| = |\mathcal{N}|^{0.5}$ nodes.

# parameters	
flat	$\|\mathcal{O}\|\|\mathcal{N}\|^2 + \|\mathcal{A}\|\|\mathcal{N}\|$
fact.	$2\|\mathcal{O}\|\|\mathcal{N}\|^{1.5} + \|\mathcal{A}\|\|\mathcal{N}\|^{0.5}$
forward-backward complexity	
flat	$O(t_{max}(\|\mathcal{N}\|\|\mathcal{S}\|^2 + \|\mathcal{N}\|^2\|\mathcal{S}\|))$
fact.	$O(t_{max}(\|\mathcal{N}\|\|\mathcal{S}\|^2 + \|\mathcal{N}\|^{1.5}\|\mathcal{S}\|))$
expectation complexity	
flat	$O(\|\mathcal{N}\|\|\mathcal{A}\|(\|\mathcal{S}\|^2 + \|\mathcal{S}\|\|\mathcal{O}\|) + \|\mathcal{N}\|^2\|\mathcal{S}\|\|\mathcal{O}\|)$
fact.	$O(\|\mathcal{N}\|\|\mathcal{A}\|(\|\mathcal{S}\|^2 + \|\mathcal{S}\|\|\mathcal{O}\|) + \|\mathcal{N}\|^{1.5}\|\mathcal{O}\|\|\mathcal{S}\| + \|\mathcal{N}\|^2\|\mathcal{O}\|)$

A 2-level factored controller with $|\mathcal{N}|^{0.5}$ nodes at each level has $2|\mathcal{O}||\mathcal{N}|^{1.5}$ parameters for $p_{n'^{top}|o'n^{base}n^{top}}$ and $p_{n'^{base}|n'^{top}o'n^{base}}$, and $|\mathcal{A}||\mathcal{N}|^{0.5}$ parameters for $p_{a|n^{base}}$. The complexity of the forward (backward) procedure is $O(t_{max}(|\mathcal{N}||\mathcal{S}|^2 + |\mathcal{N}|^{1.5}|\mathcal{S}|))$ and the complexity of computing the expectations is $O(|\mathcal{N}||\mathcal{A}|(|\mathcal{S}|^2 + |\mathcal{S}||\mathcal{O}|) + |\mathcal{N}|^{1.5}|\mathcal{O}||\mathcal{S}| + |\mathcal{N}|^2|\mathcal{O}|)$. A 2-level hierarchical controller is further restricted and therefore has fewer parameters, but the same time complexity.

5 Experiments

We first compared the performance of the maximum likelihood (ML) approach to previous optimization-based approaches from [4]. Table 2 summarizes the results for 2-layer controllers with certain combinations of $|\mathcal{N}^{base}|$ and $|\mathcal{N}^{top}|$. The problems include paint, shuttle and 4x4 maze (previously used in [4]) and three additional problems: chain-of-chains (described below), hand-washing (reduced version from [8]) and cheese-taxi (variant from [12]). On the first three problems, ML reaches the same values as the previous optimization-based approaches, but with larger controllers. We attribute this to EM's weaker ability to avoid local optima than the optimization-based approaches. However, the optimization-based approaches run out of memory on the last three problems (memory needs exceed 2 Gb of RAM), while ML scales gracefully (as analyzed in Sect. 4.2.4). ML approach demonstrates that hierarchy discovery can be tackled with tractable algorithms. We also report the values reached with a state of the art point-based value

Table 2: V^* denotes optimal values (with truncated trajectories) [3] except for handwashing and cheese-taxi where we show the optimal value of the equivalent fully-observable problem. HSVI2 found a solution in less than 1s for every problem except handwashing where the algorithm was halted after 12 hours of computation. The ML approach optimizes a factored controller for 200 EM iterations with a planning horizon of $t_{max}=100$. (5,3) nodes means $|\mathcal{N}^{base}|=5$ and $|\mathcal{N}^{top}|=3$. For cheese-taxi, we get a maximum value of 2.25. N/A indicates that the solver did not complete successfully. All tests are done on a dual-core x64 processor @2.2GHz.

Problem	$\|\mathcal{S}\|,\|\mathcal{A}\|,\|\mathcal{O}\|$	V^*	HSVI2	Best results from [4]			ML approach (avg. over 10 runs)		
			V	nodes	t(s)	V	nodes	t(s)	V
paint	4, 4, 2	3.28	3.29±0.04	(1,3)	<1	3.29	(5,3)	0.96±0.3	3.26±0.004
shuttle	8, 3, 5	32.7	32.9±0.8	(1,3)	2	31.87	(5,3)	2.81±0.2	31.6±0.5
4x4 maze	16, 4, 2	3.7	3.75±0.1	(1,2)	30	3.73	(3,3)	2.8±0.8	3.72±8e−5
chain-of-chains	10, 4, 1	157.1	157.1±0	(3,3)	10	0.0	(10,3)	6.4±0.2	151.6±2.6
handwashing	84, 7, 12	⩽1052	N/A			N/A	(10,5)	655±2	984±1
cheese-taxi	33, 7, 10	⩽5.3	2.53±0.3			N/A	(10,3)	311±14	−9±11(2.25*)

iteration method (HSVI2 [15]).

The next question is whether there are computational savings when automatically discovering a hierarchy. Recall that previous work has shown that policy optimization is simplified when a hierarchy is known a priori since the space of policies is restricted. The next experiment demonstrates that policy optimization while discovering a hierarchy can be done faster and/or yield higher value when there exists good hierarchical policies. Table 3 compares the performance when optimizing flat, hierarchical and factored controllers on chain-of-chains, hand-washing and cheese-taxi. Here, the factored and hierarchical controllers have two levels and correspond respectively to the DBNs in Fig. 3(a) and 3(b).[3] The x-axis is the number of nodes for flat controllers and the product of the number of nodes at each level for hierarchical and factored controllers. Taking the product is justified by the fact that the equivalent flat controllers of some hierarchical/factored controllers require that many nodes. The graphs in the right column of Table 3 demonstrate that hierarchical and factored controllers can be optimized faster, confirming the analysis done in Sect. 4.2.4. There is no difference in computational complexity between the strictly hierarchical and unconstrained factored architectures. Recall however that the efficiency gains of the hierarchical and factored controllers are obtained at the cost of a restricted policy space. Nevertheless, the graphs in the left column of Table 3 suggest that hierarchical/factored controllers can still find equally good policies when there exist one. Factored controllers are generally the most robust. With a sufficient number of nodes, they find the best policies on all three problems. Note that factored and hierarchical controllers need at least a number of nodes equal to the number of actions in the base layer in order to represent a policy that uses all actions.

[3]Factored controllers are hierarchical controllers where the restrictions imposed by the E_t variables are removed.

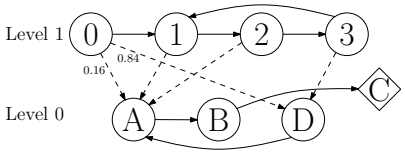
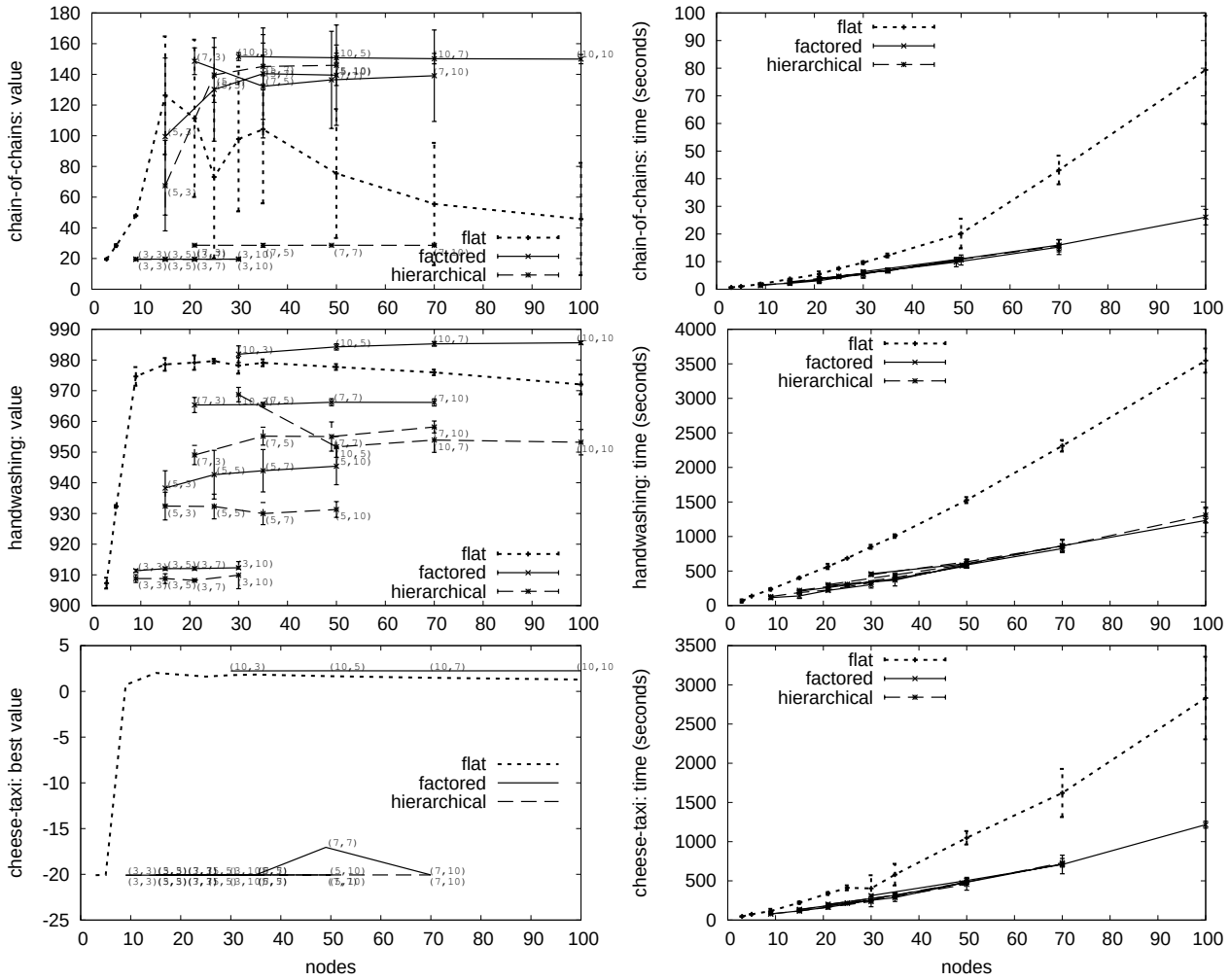


Figure 4: Hierarchical controller learnt for the chain-of-chains. The diamond indicates an exit node, for which $\hat{p}_{e^0|n^0}=1$.

This explains why hierarchical and factored controllers with less than 4 base nodes (for chain-of-chains) and 7 base nodes (for hand-washing and cheese-taxi) do poorly. The optimization of flat controllers tend to get stuck in local optima if too many nodes are used. Comparing the unconstrained factored architecture versus hierarchical, we find that the additional constraints in the hierarchical controller make the optimization problem harder although there are less parameters to optimize. As a result, EM gets stuck more often in local optima.

We also examine whether learnt hierarchies make intuitive sense. Good policies for the cheese-taxi and hand-washing problems can often be represented hierarchically, however the hierarchical policies found didn't match hierarchies expected by the authors. Since these are non-trivial problems for which there may be many ways to represent good policies in a hierarchical fashion that is not intuitive, we designed the chain-of-chains problem, which is much simpler to analyze. The optimal policy of this problem consists of executing n times the same chain of n actions followed by a submit action to earn the only reward. The optimal policy requires n^2+1 nodes for flat controllers and $n+1$ nodes at each level for hierarchical controllers. For $n=3$, ML found a hierarchical controller of 4 nodes at each level, illustrated in Fig. 4. The controller starts in node 0. Nodes at level 1 are abstract and descend into concrete nodes at level 0 by following the dashed

Table 3: *Left:* The reached values depending on the number of nodes in the controller. For the factored and hierarchical controller we indicate the number of nodes in both layers (e.g., (5,3) means $|\mathcal{N}^{base}| = 5$ and $|\mathcal{N}^{top}| = 3$) and plot the data point at $|\mathcal{N}^{base}||\mathcal{N}^{top}|$ on the x-axis. For instance, in the case of handwashing we see how the performance depends critically on $|\mathcal{N}^{base}|$. *Right:* The optimization time. In all cases, 200 EM iterations are performed with a planning horizon of $t_{max} = 100$. The results for each controller are the average of 10 runs with error bars of ± 1 standard deviation.



edges. Control is returned to level 1 when an end node (denoted by a diamond) is reached. Here, the optimal policy is to do A-B-C three times followed by D. Hence a natural hierarchy would abstract A-B-C and D into separate subcontrollers. While the controller in Fig. 4 is not completely optimal (the vertical transition from abstract node 0 should have probability 1 of reaching node A), it found an equivalent, but less intuitive abstraction by having subcontrollers that do A-B-C and D-A-B-C. This suggests that for real-world problems there will be many valid abstractions that are not easily interpretable by humans and the odds that an automated procedure finds an intuitive hierarchy without any additional guidance are slim.

6 Conclusion

The key advantage of maximum likelihood is that it can exploit the factored structure in a controller architecture. This facilitates hierarchy discovery when the hierarchical structure of the controller is encoded into a corresponding dynamic Bayesian network (DBN). Our complexity analysis and the empirical run time analysis confirm the favorable scaling. In particular, we solved problems like handwashing and cheese-taxi that could not be solved with the previous approaches in [4]. Compared to flat controllers, factored controllers are faster to optimize and less sensitive to local optima when they have many nodes. Our current implementation does not exploit any factored structure

in the state, action and observation space, however we envision that a factored implementation would naturally scale to large factored POMDPs.

For the chain-of-chains problem, maximum likelihood finds a valid hierarchy. For other problems like handwashing, there might be many hierarchies and the one found by our algorithm is usually hard to interpret. We cannot expect our method to find a hierarchy that is human readable. Interestingly, although the strictly hierarchical architectures have less parameters to optimize, they seem to be more susceptible to local optima as compared to a factored but otherwise unconstrained controller. Future work will investigate various heuristics to escape local optima during optimization.

In this paper we made explicit assumptions about the structure – we prefixed the structure of the DBN to mimic a strict hierarchy or a level-wise factorization and we fixed the number of nodes in each level. However, the DBN framework allows us to build on existing methods for structure learning of graphical models. A promising extension would be to use such structure learning techniques to optimize the factored structure of the controller. Since the computational complexity for evaluating (training) a single structure is reasonable, techniques like MCMC could sample and evaluate a variety of structures. This variety might also help to circumvent local optima, which currently define the most dominant limit of our approach.

Acknowledgments

Part of this work was completed while Charlin was at the University of Waterloo. Toussaint acknowledges support by the German Research Foundation (DFG), Emmy Noether fellowship TO 409/1-3. Poupart and Charlin were supported by grants from the Natural Sciences and Engineering Research Council of Canada, the Canada Foundation for Innovation and the Ontario Innovation Trust.

References

[1] C. Amato, D. Bernstein, and S. Zilberstein. Solving POMDPs using quadratically constrained linear programs. In *IJCAI*, pages 2418–2424, 2007.

[2] D. Braziunas and C. Boutilier. Stochastic local search for POMDP controllers. In *AAAI*, pages 690–696, 2004.

[3] A. Cassandra. *Exact and approximate algorithms for partially observable Markov decision processes*. PhD thesis, Brown University, Dept. of Computer Science, 1998.

[4] L. Charlin, P. Poupart, and R. Shioda. Automated hierarchy discovery for planning in partially observable environments. In *NIPS*, pages 225–232, 2006.

[5] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[6] E. Hansen. An improved policy iteration algorithm for partially observable MDPs. In *NIPS*, 1998.

[7] E. Hansen and R. Zhou. Synthesis of hierarchical finite-state controllers for POMDPs. In *ICAPS*, pages 113–122, 2003.

[8] J. Hoey, A. von Bertoldi, P. Poupart, and A. Mihailidis. Assisting persons with dementia during handwashing using a partially observable Markov decision process. In *ICVS*, 2007.

[9] N. Meuleau, L. Peshkin, K.-E. Kim, and L. Kaelbling. Learning finite-state controllers for partially observable environments. In *UAI*, pages 427–436, 1999.

[10] K. Murphy and M. Paskin. Linear time inference in hierarchical HMMs. In *NIPS*, 2001.

[11] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

[12] J. Pineau. *Tractable Planning Under Uncertainty: Exploiting Structure*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2004.

[13] J. Pineau, G. Gordon, and S. Thrun. Policy-contingent abstraction for robust robot control. In *UAI*, pages 477–484, 2003.

[14] P. Poupart and C. Boutilier. Bounded finite state controllers. In *NIPS*, 2003.

[15] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *UAI*, 2004.

[16] P. Stone and M. Veloso. A layered approach to learning client behaviors in the RoboCup soccer server. *Applied Artificial Intelligence*, 12:165–188, 1998.

[17] G. Theocharous, K. Murphy, and L. Pack Kaelbling. Representing hierarchical POMDPs as DBNs for multi-scale robot localization. In *ICRA*, pages 1045–1051. IEEE, 2004.

[18] M. Toussaint, S. Harmeling, and A. Storkey. Probabilistic inference for solving (PO)MDPs. Technical Report EDI-INF-RR-0934, School of Informatics, University of Edinburgh, 2006.

Modelling local and global phenomena with sparse Gaussian processes

Jarno Vanhatalo
Department of Biomedical Engineering
and Computational Science
Helsinki University of Technology
02015 TKK, Finland

Aki Vehtari
Department of Biomedical Engineering
and Computational Science
Helsinki University of Technology
02015 TKK, Finland

Abstract

Much recent work has concerned sparse approximations to speed up the Gaussian process regression from the unfavorable $O(n^3)$ scaling in computational time to $O(nm^2)$. Thus far, work has concentrated on models with one covariance function. However, in many practical situations additive models with multiple covariance functions may perform better, since the data may contain both long and short length-scale phenomena. The long length-scales can be captured with global sparse approximations, such as fully independent conditional (FIC), and the short length-scales can be modeled naturally by covariance functions with compact support (CS). CS covariance functions lead to naturally sparse covariance matrices, which are computationally cheaper to handle than full covariance matrices. In this paper, we propose a new sparse Gaussian process model with two additive components: FIC for the long length-scales and CS covariance function for the short length-scales. We give theoretical and experimental results and show that under certain conditions the proposed model has the same computational complexity as FIC. We also compare the model performance of the proposed model to additive models approximated by fully and partially independent conditional (PIC). We use real data sets and show that our model outperforms FIC and PIC approximations for data sets with two additive phenomena.

1 Introduction

Gaussian processes (GP) are powerful tools for Bayesian nonlinear and nonparametric regression. They can be used as prior for underlying latent function, on which the observations are conditioned (Rasmussen and Williams, 2006). The main limitation with GP models is their unfavorable $O(n^3)$ scaling in training time and $O(n^2)$ in memory, where n is the size of the training set. In recent years, much research has concerned sparse approximations to speed up the computations down to $O(nm^2)$ and reduce the memory requirements to $O(nm)$, with $m \ll n$ (e.g. Snelson and Ghahramani, 2006, 2007; Lawrence, 2003; Seeger et al., 2003; Williams and Seeger, 2001; Quiñonero-Candela and Rasmussen, 2005). Another approach to speed up training and save memory is to use GPs with compactly supported (CS) covariance functions. These are special kinds of functions that construct naturally sparse covariance matrices (e.g. Wendland, 2005; Rasmussen and Williams, 2006; Storkey, 1999).

In this paper, we will treat both the sparse GP approximations and GPs with naturally sparse covariance function. These two concepts are rather different, but share the property that they are computationally more efficient than full GPs. The sparse approximations considered here are fully and partially independent conditional (FIC/PIC) (Snelson and Ghahramani, 2006, 2007). The naturally sparse GP is discussed for a piecewise polynomial covariance function. FIC provides a global approximation that performs well with rather long length-scales. PIC combines both local and global type of approximations and is able to model also short length-scale phenomena. The problem with PIC is, however, that it introduces discontinuities in the correlation structure. The CS covariance functions, on the other hand, provide a natural way to model the local phenomena without anomalies in the correlation structure.

In many practical problems, it is plausible that the underlying function combines both long and short length-scale phenomena. In this case, we can construct an additive GP model with two covariance functions. To our best knowledge, the sparse GP literature thus far has

concentrated on modelling either long or short length-scale phenomena. The purpose of this work is to add these two concepts together to construct a sparse GP model that is able to capture both local and global properties at the same time. Therefore, we propose to add up the covariance function induced by FIC approximation with a CS covariance function so that the long length-scale phenomena are captured by FIC and the local variations by CS function. We will apply the model for real data sets and compare the results to additive full GP models approximated by FIC and PIC. We will show that our method outperforms the approximations in performance and is computationally of same complexity under certain conditions.

2 Gaussian process regression

We will consider a regression problem, where we have scalar observations $\mathbf{y} = \{y_i\}_{i=1}^n$ at input locations $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ and the observations are assumed to satisfy

$$y_i = g(\mathbf{x}_i) + \epsilon, \text{ where } \epsilon \sim \mathcal{N}(0, \sigma^2). \quad (1)$$

Each of the input vectors $\mathbf{x}_i$ is of dimension D, which is assumed to be rather low (say $D \leq 3$), and g is a nonlinear latent function, which we are interested in. The latent function is given a Gaussian process prior, which implies that any finite subset of latent variables, $\mathbf{g} = \{g(\mathbf{x}_i\}_{i=1}^n$, has a multivariate Gaussian distribution. In particular, at the observed input locations $\mathbf{X}$ the latent variables have distribution $p(\mathbf{g}|\mathbf{X}) = \mathcal{N}(\mathbf{g}|\mu, \mathbf{K}_{n,n})$, where $\mathbf{K}_{n,n}$ is the covariance matrix and μ the mean function. In this paper, without loss of generality, we will use a zero-mean Gaussian processes. The covariance matrix is constructed from a covariance function k, which represents the prior assumptions of the smoothness of the latent function. Each element in the covariance matrix is evaluated as $[\mathbf{K}_{n,n}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j, \theta)$ and a widely used covariance function is the stationary squared exponential

$$k_{\text{se}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_{\text{se}}^2 \exp\left(-\sum_{d=1}^{D}(x_{i,d} - x_{j,d})^2/l_d^2\right), \quad (2)$$

where σ_{se}^2 is the scaling parameter and l_d is the length-scale that governs how fast the correlation drops among direction d. From this on we denote the hyperparameters by $\theta = \{\sigma_{\text{se}}^2, l_1, ..., l_D, \sigma^2\}$

Since the noise model, or the likelihood, is Gaussian $p(\mathbf{y}|\mathbf{g}, \mathbf{X}, \theta) = \mathcal{N}(\mathbf{y}|\mathbf{g}, \sigma^2\mathbf{I})$, we are able to marginalise over the latent variables to obtain the marginal likelihood

$$p(\mathbf{y}|\mathbf{X}, \theta) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_{n,n} + \sigma^2\mathbf{I}). \quad (3)$$

After placing a prior for the hyperparameters $p(\theta)$, we can find the maximum a posteriori (MAP) estimate $\hat{\theta}$ for the hyperparameters by maximising the log posterior cost function,

$$\hat{\theta} = \arg\max_{\theta}\left[\log p(\theta) - \frac{1}{2}\log|\mathbf{K}_{y,y}| - \frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{K}_{y,y}^{-1}\mathbf{y}\right], \quad (4)$$

where $\mathbf{K}_{y,y} = \mathbf{K}_{n,n} + \sigma^2\mathbf{I}$. Optimising the hyperparameters can also be considered as a model selection, since we are fixing our GP model by setting the hyperparameters in their MAP estimate.

Conditioning on the data and the hyperparameter values, the posterior predictive distribution of the latent variables $g(\mathbf{X}_*)$ at new input locations $\mathbf{X}_*$ is Gaussian

$$p(\mathbf{g}_*|\mathbf{X}_*, \mathcal{D}, \hat{\theta}) = \mathcal{N}(\mathbf{g}_*|\mu_*, \mathbf{\Sigma}_{*,*}), \quad (5)$$

where $\mathcal{D} = \{\mathbf{y}, \mathbf{X}\}$ and

$$\begin{aligned}\mu_* &= \mathbf{K}_{*,n}\mathbf{K}_{y,y}^{-1}\mathbf{y} \\ \mathbf{\Sigma}_{*,*} &= \mathbf{K}_{*,*} - \mathbf{K}_{*,n}\mathbf{K}_{y,y}^{-1}\mathbf{K}_{n,*}. \end{aligned} \quad (6)$$

In many practical situations one length-scale per input dimension is too restrictive. Consider, for example, time series data, which evolves rather smoothly between different years, but at the same time has faster, monthly, variations (see figure 1). In this case, a more reasonable model is

$$y_i = g(\mathbf{x}_i) + \epsilon = f(\mathbf{x}_i) + h(\mathbf{x}_i) + \epsilon, \quad (7)$$

where the latent function g is replaced by a sum of two functions, of which the other is slowly and the other fast varying. We can now place a Gaussian process prior for both of the functions f and h and give them different covariance functions that reflect our beliefs about their smoothness properties. The sum of two Gaussian variables is Gaussian and the prior for the additive model is $p(\mathbf{g}|\mathbf{X}) = \mathcal{N}(\mathbf{g}|\mathbf{0}, \mathbf{K}_{n,n}^{(h)} + \mathbf{K}_{n,n}^{(f)})$. The marginal likelihood and posterior predictive distribution are as before with $\mathbf{K}_{n,n} = \mathbf{K}_{n,n}^{(h)} + \mathbf{K}_{n,n}^{(f)}$. However, if we are interested on only, say, phenomenon f, we can consider the h part of the latent function as correlated noise and evaluate the predictive distribution $p(f|\mathbf{X}_*, \mathcal{D}, \hat{\theta}) = N(\mathbf{K}_{*,n}^{(f)}\mathbf{K}_{y,y}^{-1}\mathbf{y}, \mathbf{K}_{*,*}^{(f)} - \mathbf{K}_{*,n}^{(f)}\mathbf{K}_{y,y}^{-1}\mathbf{K}_{n,*}^{(f)})$.

The training of the hyperparameters is conducted via gradient based optimisation. The computationally most demanding part is the evaluation of the gradient of the log marginal likelihood

$$\begin{aligned}\frac{\partial}{\partial\theta}\log p(\mathbf{y}|\mathbf{X}, \theta) =& \frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{K}_{y,y}^{-1}\frac{\partial\mathbf{K}_{y,y}}{\partial\theta}\mathbf{K}_{y,y}^{-1}\mathbf{y} \\ &- \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_{y,y}^{-1}\frac{\partial\mathbf{K}_{y,y}}{\partial\theta}\right). \end{aligned} \quad (8)$$

The matrix inversion and the determinant in marginal likelihood and its gradient scale as $O(n^3)$ in time. Together with $O(n^2)$ memory requirements this prevents the direct implementation of GP for large problems.

3 FIC and PIC sparse approximations

In this section, we consider the fully and partially independent conditional sparse approximations for GP. We will give a short review of the methods, but readers interested in detailed derivation of the approximations should refer to the original papers by Snelson and Ghahramani (2006, 2007), or for a more general perspective to Quiñonero-Candela and Rasmussen (2005).

The approximations are based on introducing an additional set of latent variables $\mathbf{u} = \{u_i\}_{i=1}^m$, called *inducing variables*, that correspond to a set of input locations $\mathbf{X}_u$, *inducing inputs*. The prior on latent function is approximated by

$$p(\mathbf{g}|\mathbf{X}) \approx q(\mathbf{g}|\mathbf{X},\mathbf{X}_u) = \int q(\mathbf{g}|\mathbf{X},\mathbf{X}_u,\mathbf{u})p(\mathbf{u}\,|\mathbf{X}_u)\mathrm{d}\mathbf{u}, \quad (9)$$

where $\mathbf{g}$ is conditionally dependent on $\mathbf{u}$ through the inducing conditional $q(\mathbf{g}|\mathbf{X}, \mathbf{X}_u, \mathbf{u})$. In the FIC framework, the latent variables are conditionally independent given $\mathbf{u}$. In this case, the inducing conditional factorises into $q(\mathbf{g}|\mathbf{X}, \mathbf{X}_u, \mathbf{u}) = \prod_{i=1}^n q_i(g_i|\mathbf{X}, \mathbf{X}_u, \mathbf{u})$. In contrast, in PIC the latent variables are set into blocks that are conditionally independent, but within a block they are fully dependent. The approximate conditionals of FIC and PIC can be summarised as

$$q(\mathbf{g}|\mathbf{X}_u, \mathbf{u}) = \mathcal{N}(\mathbf{K}_{n,u}\mathbf{K}_{u,u}^{-1}\,\mathbf{u}, \mathrm{mask}\,(\mathbf{K}_{n,n} - \mathbf{Q}_{n,n}, \mathbf{M})),$$

where $\mathbf{Q}_{a,b} = \mathbf{K}_{a,u}\mathbf{K}_{u,u}^{-1}\mathbf{K}_{u,b}$ and $\mathbf{K}_{n,u}$ is the covariance between latent variables and the inducing variables. The function $\mathrm{mask}\,(\mathbf{K}, \mathbf{M})$, with matrix $\mathbf{M}$ of ones and zeros, returns a matrix $\mathbf{\Lambda}$ of size $\mathbf{M}$ and elements $\mathbf{\Lambda}_{ij} = \mathbf{K}_{ij}$ if $\mathbf{M}_{ij} = 1$ and zero otherwise. An approximation with $\mathbf{M} = \mathbf{I}$ corresponds to FIC and an approximation, where $\mathbf{M}$ is block diagonal corresponds to PIC.

By placing a zero-mean Gaussian prior on the inducing variables, $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{u,u})$, and integrating over them we get the approximate prior over latent variables

$$q(\mathbf{g}|\mathbf{X}, \mathbf{X}_u) = \mathcal{N}(\mathbf{0}, \mathbf{Q}_{n,n} + \mathbf{\Lambda}). \quad (10)$$

Here the inducing inputs can be considered as an additional set of hyperparameters. The marginal likelihood is $p(\mathbf{y}\,|\mathbf{X}, \mathbf{X}_u, \theta) = \mathcal{N}(\mathbf{0}, \mathbf{Q}_{n,n} + \mathbf{\Lambda} + \sigma^2\mathbf{I})$, where the covariance is now a sum of the rank m matrix $\mathbf{Q}_{n,n}$ and (block)diagonal matrix $\mathbf{\Lambda} + \sigma^2\mathbf{I}$. This leads to computational savings in training, since the crucial computations only require $O(nm^2)$. Other major advantage is the save in memory requirements, since the sparse approximations need only $O(nm)$ memory compared to $O(n^2)$ in full GP.

In the literature on FIC and PIC, the approximations are used for the one covariance function regression problem (1). The results show that the approximations are efficient for a wide variety of such problems (Snelson, 2007). However, the two covariance function regression problem (7) is somewhat trickier as will be seen in the experiments. With FIC all the correlations are induced through the inducing inputs. Thus, modelling very short length-scale phenomena becomes unpractical, since one would need infeasible many inducing inputs to capture fast variations. PIC, on the other hand, can capture short length-scales within a block, but its shortcoming is that between the blocks latent variables are independent. This leads to discontinuities in correlation structure and reduces the predictive performance on the block boundaries.

4 Gaussian process with compact support covariance function

In this section, we will consider Gaussian processes with naturally sparse covariance matrix. We will first give short introduction on compact support covariance function, after which we consider their implementation issues within GP framework.

4.1 Compact support

With a compactly supported covariance function we mean a function that gives zero correlation between data points whose distance exceeds a certain threshold. Such functions form naturally sparse covariance matrices, that give savings in computational time and memory requirements compared to full GPs. The challenge in constructing CS covariance functions is to guarantee their positive definiteness. A full support covariance function can not be cut arbitrarily to obtain a compact support, since the resulting function would not, in general, be positive definite. One option is to use a family of piecewise polynomial functions as, for example,

$$k_{\mathrm{pp}} = \frac{\sigma^2_{\mathrm{pp}}}{3}(1-r)_+^{j+2}\big((j^2+4j+3)r^2+(3j+6)r+3\big), \quad (11)$$

where $j = \lfloor D/2 \rfloor + 3$ and $r^2 = \sum_{d=1}^D (x_{i,d} - x_{j,d})^2/l_d^2$ (see e.g. Wendland, 2005; Rasmussen and Williams, 2006). The function k_{pp} is positive definite up to the input dimensions D. There are many other CS covariance functions and, for example, Gneiting (2002) proposed a method that constructs a CS covariance function that preserves the smoothness properties of powered exponential and Matern class of functions.

In this work, we concentrate on, how GP analysis can efficiently be done with CS functions and leave the comparison of different functions for future. The CS covariance functions in relation to GPs have been studied already by, for example, Storkey (1999). He considered covariance matrices of Toeplitz form, which are

fast to handle due their banded structure. However, constructing Toeplitz covariance matrices is not possible in two or higher dimensions without approximations. Here we will present an approach that works for any kind of sparse covariance matrix.

The computationally intensive parts in the cost function (4) and its derivatives (8) are the determinant and multiplication of matrix or vector by the inverse covariance matrix. Albeit the covariance matrix is sparse its inverse is usually full and requires time $O(n^3)$ to compute. Fortunately, we are able to do all these calculations without forming the full inverse. The key role is played by the Cholesky factorisation of a covariance matrix.

4.2 The computations

The Cholesky factorization $\mathbf{L}$ of symmetric positive definite matrix $\mathbf{A}$ is a lower triangular matrix such that $\mathbf{A} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$. For full matrix, it can be found in time $O(n^3)$. For sparse matrices, however, this is faster since the sparsity is preserved in the Cholesky factorisation. The factorisation time depends on the sparsity structure of the matrix $\mathbf{A}$. It can be reduced by permuting the columns and rows of matrix so that the number of non-zero elements in $\mathbf{L}$ are minimised (for example, using minimum degree ordering or nested dissection (Amestoy et al., 2004; Davis, 2006; Rue and Martino, 2007)). The time needed for permutation is, in general, negligible compared to the time needed for other evaluations with GP. For example, in $N \times N$ lattices and with small length-scale in CS function the typical cost of factorising CS covariance matrix after permutation is $O(n^{3/2})$ for $n = N^2$ and for $N \times N \times N$ lattices $O(n^2)$, for $n = N^3$. After finding the Cholesky decomposition of matrix $\mathbf{A}$, we can efficiently evaluate its log determinant and $\mathbf{y}^{\mathrm{T}}\mathbf{A}\mathbf{y}$ ($O(n)$ and $O(n \log n)$ operations with 2D lattice data).

The term that needs most concern is $\mathrm{tr}(\mathbf{K}_{y,y}^{-1}\partial\mathbf{K}_{y,y}/\partial\theta)$ in the gradient of the marginal likelihood (8). The matrix $\mathbf{B} = \partial\mathbf{K}_{y,y}/\partial\theta$ has the same sparsity structure as $\mathbf{K}_{y,y}$. Thus, if we denote $\mathbf{Z} = \mathbf{K}_{y,y}^{-1}$ we can write $\mathrm{tr}(\mathbf{Z}\mathbf{B}) = \mathrm{tr}(\mathbf{Z}_{\mathrm{sp}}\mathbf{B})$, where $\mathbf{Z}_{\mathrm{sp}}$ is a sparse representation of $\mathbf{Z}$, which has non-zero elements only where $\mathbf{B}_{ij} \neq 0$. We can obtain such a matrix by using an algorithm introduced by Takahashi et al. (1973).

The algorithm is derived as follows. First, determine the Cholesky decomposition of $\mathbf{K}_{y,y}$ and write

$$\mathbf{L}^{\mathrm{T}}\mathbf{Z} = \mathbf{L}^{-1}. \tag{12}$$

Next, take the diagonal of the Cholesky triangle, $\mathbf{D} = \mathrm{mask}(\mathbf{L}, \mathbf{I})$, write the equation (12) as

$$\mathbf{D}\mathbf{Z} + (\mathbf{L}^{\mathrm{T}} - \mathbf{D})\mathbf{Z} = \mathbf{L}^{-1}, \tag{13}$$

subtract the second term on the left hand side and multiply by $\mathbf{D}^{-1}$ from left

$$\mathbf{Z} = \mathbf{D}^{-1}\mathbf{L}^{-1} - \mathbf{D}^{-1}(\mathbf{L}^{\mathrm{T}} - \mathbf{D})\mathbf{Z}. \tag{14}$$

Now, since the inverse is symmetric, we can give a recursive formula for the elements of the inverse as

$$\mathbf{Z}_{ij} = \frac{\delta_{ij}}{\mathbf{D}_{ii}^2} - \frac{1}{\mathbf{D}_{ii}} \sum_{k=i+1}^{n} \mathbf{L}_{ki}\mathbf{Z}_{kj}, \;\; j \geq i, i = n, ..., 1. \tag{15}$$

We can find the upper triangle of the inverse by looping i from n to 1 and j from n to i. The lower triangular of $\mathbf{Z}$ can then be filled according to symmetry. To find the sparse inverse we evaluate only a small fraction of the elements.

Let us denote by $\mathbf{C}$ an adjacency matrix, which has $\mathbf{C}_{ij} = 1$ if $\mathbf{L}_{ij} \neq 0$ or $\mathbf{L}_{ij}^{\mathrm{T}} \neq 0$ and zero otherwise. Here we consider $\mathbf{L}_{ij}$ non-zero even if its numerical value was zero, but it is symbolically non-zero (that is, it has to be evaluated, when solving for sparse Cholesky factorisation (Davis, 2006)). To find the sparse inverse, we need to evaluate only the elements $\mathbf{Z}_{ij}$ such that $\mathbf{C}_{ij} \neq 0$. To justify this, consider the recursive formula (15). The algorithm needs at least all the elements $\{\mathbf{Z}_{ij} : \mathbf{C}_{ij} \neq 0\}$, since these are the elements that are needed for the diagonal of $\mathbf{Z}$. From the Cholesky factorisation it follows that $\mathbf{C}_{ij}$ is non-zero, if for any $k < i, j$ the elements $\mathbf{C}_{ik}$ and $\mathbf{C}_{kj}$ are non-zero:

$$k < i, j \;\; \mathbf{C}_{ik} \neq 0 \;\; \mathbf{C}_{kj} \;\; \neq 0 \Rightarrow \;\; \mathbf{C}_{ij} \neq 0.$$

This property ensures that using the equation (15) we are able to find all the elements $\mathbf{Z}_{ij}$, for which $\mathbf{C}_{ij} \neq 0$, by looping i from n to 1 and at each i evaluating the elements $\mathbf{Z}_{ij}, j \in c_i = \{j \geq i, C_{ij} \neq 0\}$. The algorithm is discussed in detail by Niessner and Reichert (1983).

4.3 Computational complexity

The computational complexity and memory requirements of CS+FIC depend on the number of (symbolically) nonzero elements in the Cholesky factorisation of $\mathbf{K}_{y,y}$. This is always equal or more than the number of non-zeros in the covariance matrix and with 2D and 3D lattice data it is $O(n \log n)$ and $O(n^{4/3})$ (Davis, 2006). If γ_k denotes the number of non-zero elements outside the diagonal of column k of $\mathbf{L}$, then the computational cost for finding the sparse inverse is $O(\sum_{k=1}^{n} \gamma_k(\gamma_k + 1))$. To get an intuition of the computation time we can consider banded and full covariance matrices as limiting cases. Finding the sparse inverse of banded covariance matrix needs time $O(n(b/2)^2)$, where b denotes the band width, and finding the full inverse is $O(n^3/3)$ operation. The other sparse inverses place between these. For example, the average

γ_k of the two and three dimensional lattice data are $O(\log n)$ and $O(n^{1/3})$. Thus the recursion takes time $O(n \log(n)^2)$ and $O(n^{5/3})$, respectively. For randomly distributed data points this is an approximation.

5 Additive model with CS covariance function and FIC

Here, we propose a new sparse GP model that combines the ideas behind sparse approximations and CS covariance functions.

5.1 Additive sparse GP model

The covariance matrix in the FIC approximate prior (10) can be interpreted as a realisation of special kind of covariance function $k_{\mathrm{FIC}} = f(k(\mathbf{x}_i, \mathbf{x}_j, \theta), \mathbf{X}_u)$. This is a function of the original covariance function, its hyperparameters and the inducing inputs. By adding up the FIC and CS covariance functions we are able to construct a sparse GP model for the two component regression problem (7) with prior

$$p(\mathbf{g}|\mathbf{X}, \mathbf{X}_u, \theta) = \mathcal{N}(\mathbf{0}, \mathbf{Q}_{n,n} + \mathbf{\Lambda} + \mathbf{K}_{n,n}^{(\mathrm{CS})}). \qquad (16)$$

We will refer to this later as CS+FIC model. Here, the matrix $\mathbf{Q}_{n,n}$ is of rank m and the matrix $\mathbf{\Lambda} + \mathbf{K}_{n,n}^{(\mathrm{CS})}$ is sparse with the same sparsity structure as in $\mathbf{K}_{n,n}^{(\mathrm{CS})}$. Now, we can conduct the training in similar manner to FIC and PIC by using the Woodbury-Sherman-Morrison lemma

$$(\mathbf{Q}_{n,n} + \hat{\mathbf{\Lambda}})^{-1} = \hat{\mathbf{\Lambda}}^{-1} - \mathbf{V}\mathbf{V}^{\mathrm{T}}, \qquad (17)$$

where

$$\mathbf{V} = \hat{\mathbf{\Lambda}}^{-1}\mathbf{K}_{g,u}\mathrm{chol}[(\mathbf{K}_{u,u} + \mathbf{K}_{u,g}\hat{\mathbf{\Lambda}}^{-1}\mathbf{K}_{g,u})^{-1}].$$

In case of PIC and FIC $\hat{\mathbf{\Lambda}} = \mathbf{\Lambda} + \sigma^2\mathbf{I}$ and in CS+FIC $\hat{\mathbf{\Lambda}} = \mathbf{K}_{n,n}^{(\mathrm{CS})} + \mathbf{\Lambda} + \sigma^2\mathbf{I}$. In FIC and PIC, most of the time is spent in the matrix multiplications in $\mathbf{V}$. The inverse of $\mathbf{\Lambda} + \sigma^2\mathbf{I}$ can be evaluated in $O(n)$, with FIC, and in $O(nb^2)$, with PIC, where b is the blocksize. After this, the rest of the computations required in the log marginal likelihood and its gradients involve sums and products of $\hat{\mathbf{\Lambda}}^{-1}$ and $\mathbf{V}$ with (block)diagonal matrices and matrices of size at most $n \times m$ (for technical details see, e.g., Snelson (2007))[1].

5.2 Computational issues

The matrix multiplications between $n \times m$ matrices are present also in CS+FIC model, and, thus, its computational time is at least the same as for FIC and PIC. The interesting question, however, is what is the upper bound for the computational complexity of CS+FIC. The terms that need consideration are those involving $\hat{\mathbf{\Lambda}}$. The multiplications $\hat{\mathbf{\Lambda}}^{-1}\mathbf{P}$, where $\mathbf{P}$ is an $n \times m$ matrix (for example, $\mathbf{K}_{g,u}$ in $\mathbf{V}$), are computed by solving m linear equations, and the terms $\mathrm{tr}(\hat{\mathbf{\Lambda}}^{-1}\partial\mathbf{H}/\partial\theta)$, where $\mathbf{H}$ has the same sparsity structure as $\hat{\mathbf{\Lambda}}$, are solved using the sparse inverse algorithm described earlier. If we assume uniformly distributed 2D lattice data and small length-scale for CS function, we need $O(n^{3/2})$ time for the Cholesky factorisation of $\hat{\mathbf{\Lambda}}$, $O(nm \log(n))$ for solving the m linear equations and $O(n \log(n)^2)$ for the sparse inverse. With three dimensional lattice, the computations are governed by the $O(n^2)$ scaling of the Cholesky factorisation. We can now conclude that the computational complexity of the CS+FIC model is: $O(nm^2) \leq O_{\mathrm{CS+FIC}} \leq O(\max(nm^2, n \log(n)^2, n^{3/2}))$ for 1D and 2D lattice. For the 3D lattice the performance is $O(nm^2) \leq O_{\mathrm{CS+FIC}} \leq O(\max(nm^2, n^2))$. The memory requirements are $O(n \log(n))$ and $O(n^{4/3})$ respectively. Thus, if $m \geq \log(n)$ and $m \geq \sqrt{n}$ the training of CS+FIC model scales up to a constant as the training of FIC (For example 100 inducing inputs and 10 000 data points).

6 Experiments

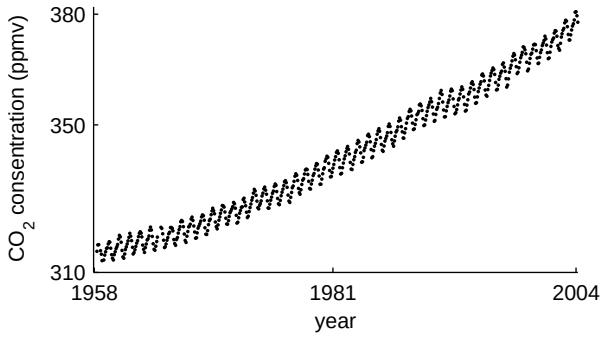
In this section, we present results on applying the models discussed above to several data sets.

6.1 Maunaloa CO_2 data

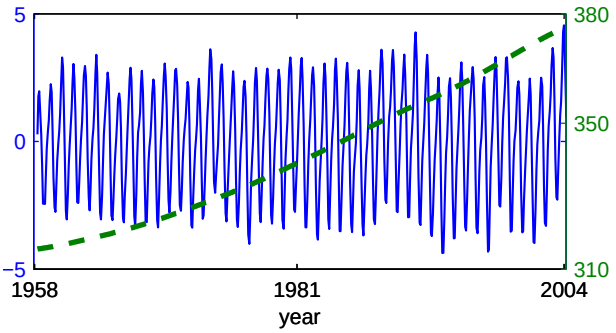
The data contain the atmospheric CO_2 concentrations (ppmv) collected at Mauna Loa Observatory, Hawaii, every month from 1958 until 2004[2] (see figure 1(a)). The data consist of 557 points (seven measurements were missing) and were previously analysed, for example, by Rasmussen and Williams (2006). They presented a sophisticated model that was able to capture the periodicity, long term trend and changes in the periodicity. Here, the aim is to test sparse additive GP models. The models compared are: full GP, FIC and PIC with $k_{\mathrm{se}} + k_{\mathrm{pp}}$ covariance function, and CS+FIC model, where the CS part is given by k_{pp} and FIC part uses k_{se}. We placed 24 inducing inputs in a regular grid over the input space. Since FIC failed with 24 inducing inputs, we ran additional experiments with 141 inducing inputs with FIC. The blocks for PIC were set regularly so that the number of data points in a block was approximately same as the number of inducing inputs. The length-scales were given half Students'-t prior with 3 degrees of freedom and variance 4 and

[1] The Matlab implementation of the models used in this work will be published in www.lce.hut.fi/research/mm/gp/

[2] http://cdiac.esd.ornl.gov/ftp/trends/co2/maunaloa.co2



(a) The Maunaloa CO_2 data.



(b) The posterior predictive mean of the monthly and overall trend with CS+FIC. On the left the scale for the short term trend and on the right for the long term trend.

Figure 1: The Maunaloa CO_2 data and predictions with CS+FIC model.

the magnitudes half Students'-t with 0.3 degrees of freedom and variance 4.

The predictive mean of the CS+FIC model for the short and long term behaviour are shown in the figure 1(b) and the model performances are presented in the table 1. We conducted 10-fold cross-validation and evaluated the root mean squared error (RMSE) and the mean log predictive density (MLPD) for each of the models. The CS+FIC was as good as the full GP in both of the performance tests. PIC was little worse than CS+FIC, which can be explained by the discontinuities on the block boundaries. FIC with 24 inducing inputs did orders of magnitude worse than all the other models, because it was able to capture only the long trend phenomenon and considered the monthly changes as noise. With 141 inducing inputs FIC did better, since it was able to model also the monthly changes. However, it was still worse than the other models.

6.2 The US annual precipitation data

The US precipitation data consist of monthly precipitation measures recorded across the whole country

Table 1: The model performances in the Maunaloa CO_2 data obtained with 10-fold cross validation.

Model	RMSE	MLPD
CS+FIC (m=24)	0.317	-0.251
full GP	0.316	-0.250
FIC (m=24)	2.151	-2.189
FIC (m=141)	0.83	-1.265
PIC (m=24)	0.401	-0.318

from 1895 to 1997[3]. The data consist of the spatial co-ordinates and elevation of the stations, and the precipitation in millimeters per month. In total, there are 11918 stations, but a high fraction of the measurements are missing. For the analysis in this paper, we collected the stations that recorded all the measurements for the year 1995. This leads to total 5776 stations (the station locations are shown in figure 2(a)). The data was previously used by Paciorek and Schervish (2006), who studied the subregion of Colorado with non-stationary covariance function. However, they recorded equally good results for stationary covariance functions, which suggests that an additive stationary model could work reasonably well also for the whole country.

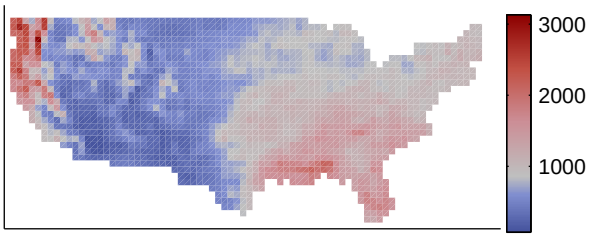
We will study the data first with a model that uses only the spatial co-ordinates as inputs for GP. After this we study the same data with all three explanatory variables. Since the number of data points is infeasible high for full GP, we tested only FIC, PIC and CS+FIC with the same covariance functions as in the previous experiment. The inducing inputs were treated in two different ways. First, they were initialised by picking 90 data points (for FIC we used also 225 inducing inputs) randomly from the data and optimized. This gave equally good results compared to the model, where they were placed in regular lattice over the country. Thus, we will report only the results from the latter scheme. The blocks for the PIC model were placed regularly over the country so that the number of data points inside each block was approximately the same as the number of inducing inputs. For the training, the input co-ordinates were scaled between 1-120 and the targets between -9-35. The priors for hyperparameters were the same as in the previous experiment.

The model performances were evaluated via 10-fold cross validation. Results are shown in the tables 2 and 3. The posterior predictive mean of the precipitation levels across the country is shown in figure 2(b). Again in this data set, CS+FIC outperforms the FIC

[3]http://www.image.ucar.edu/GSP/Data/US.monthly.met/



(a) The station locations.



(b) The annual precipitation in the US in 1995.

Figure 2: The US annual precipitation data. The upper figure shows the data points and the lower figure shows the CS+FIC posterior mean of the annual precipitation in the US in 1995.

Table 2: The model performances in the US annual precipitation data with *spatial covariates* obtained with 10-fold cross validation.

Model	RMSE	MLPD
CS+FIC (m=90)	208.3	-2.164
FIC (m=90)	278.1	-2.444
FIC (m=225)	263.0	-2.394
PIC (m=90)	210.3	-2.175

and PIC models, but not as clearly as in Maunaloa dataset. The precipitation level varies rather smoothly across most of the US and, thus, FIC and PIC are able to capture the behaviour in large areas in the east and middle of the country. The quick changes in the precipitation level take place in the mountainous regions in the west. These changes can be explained by the elevation information, since the models with elevation information have considerably better predictive performance than the models without.

6.3 Computational performance

In the Maunaloa data set, the training time of CS+FIC was the same as the training time of FIC/PIC. In this case the data was collected with constant rate, which resulted in banded (Toeplitz form) $\hat{\mathbf{\Lambda}}$, whose Cholesky

Table 3: The model performances in the US annual precipitation data with *spatial and elevation* covariates obtained with 10-fold cross validation.

Model	RMSE	MLPD
CS+FIC (m=90)	165.2	-1.788
FIC (m=90)	249.1	-2.321
FIC (m=225)	216.9	-2.189
PIC (m=90)	168.7	-1.898

factorisation is very sparse. In the precipitation data sets, CS+FIC was little slower than FIC/PIC, but still the training time was only few minutes with standard office PC. The matrix $\hat{\mathbf{\Lambda}}$ had less than 3% non-zero elements in all the data sets and the memory requirements of CS+FIC were the same as in FIC/PIC.

The scaling of the training time of CS+FIC was tested by keeping the amount of inducing inputs and data points constant at turn and altering the other. With Maunaloa data set CS+FIC scaled exactly as FIC and PIC. With the precipitation data sets the training time of CS+FIC with fixed number of inducing inputs increased slightly faster than theoretically. The reasons for this are non uniformly collected data and the extra overhead in the sparse inverse algorithm for keeping track of the non-zero elements in $\mathbf{Z}_{\text{sp}}$. The sparse inverse algorithm is currently implemented with Matlab, which results in rather high overhead in the for-loops. Also, the short length-scale phenomena might not have been present in the subsampled data sets. This increases the sparsity of $\hat{\mathbf{\Lambda}}$ and speeds up the CS+FIC model. When comparing the computation times, one has to remember that in contrast to FIC/PIC the training time of CS+FIC depends on the characteristics of the data. Thus, direct comparison of the training times is hard.

Albeit we have considered only low dimensional problems, nothing prevents us to ably CS+FIC for high dimensional problems as well. In general, the density of the sparse covariance matrix tends to increase as the input dimension gets higher. If the data are evenly distributed over the space, the number of data points covered by constant length-scale increases together with dimensionality. However, in order to find fast varying phenomena the data has to be densely distributed across all dimensions. High dimensional regression problems often seem to have broad trends, partly because the density of the data is so low that the fast phenomena can not be found. On request of the reviewers we tested our method also in two data sets, with higher dimensionality than three. The data sets were forest fires data[4], with 12 inputs and 517

[4]http://archive.ics.uci.edu/ml/datasets/Forest+Fires

data points, and add10 data[5], with ten inputs and 5492 data points. The forest fires data did not contain additive phenomena and the model performance was equally good with all the models. The training times were the same as in Maunaloa data. The add10 data consists of additive components that are functions of different inputs. In this data the CS+FIC worked slightly better and was as fast as in the US precipitation data sets. Due to space constraints the exact results were left out.

7 Conclusions

In this paper, we introduced an additive sparse Gaussian process that can model both long and short length-scale phenomena in the data. The model consists of FIC sparse approximation, which is used for the global phenomenon, and a compact support covariance function to model the local behavior. The analysis of the proposed model is conducted using sparse matrix routines and sparse inverse algorithm introduced by Takahashi et al. (1973). Under certain conditions the computational time is shown to scale similarly to the training time of FIC and PIC.

The CS+FIC model was compared to FIC and PIC approximations with several data sets. We found that the proposed model gives better overall performance than FIC and PIC, if there are two additive phenomena in the data, and equally good performance in non-additive data sets. Approximating additive GP models with short length-scale phenomena by FIC leads to poor performance, because the approximation is global by its nature. PIC models rather well also short length-scales, but its short coming are the discontinuities in the correlation structure. Our model combines the good global properties of FIC and the good local properties of compact support covariance function.

In practical problems it is often reasonable to use additive models for the purposes of data analysis. The proposed CS+FIC model provides a practical tool for modeling large data sets, which are infeasible for full Gaussian processes. The CS+FIC model is faster and requires less memory than full GP. It is also more accurate than FIC and PIC in additive models.

Acknowledgments

Authors thank Marc Deisenroth for excellent comments on the paper. The first author thanks also the Graduate School in Electronics, Telecommunications and Automation (GETA) and Finnish Funding Agency for Technology and Innovation for funding his post graduate studies and this research.

[5]http://www.cs.toronto.edu/ delve/data/add10/desc.html

References

Amestoy, P., Davis, T. A., and Duff, I. S. (2004). Algorithm 837: AMD, an approximate minimum degree ordering algorithm. *ACM Transactions on Mathematical Software*, 30(3):381–388.

Davis, T. A. (2006). *Direct Methods for Sparse Linear Systems*. SIAM.

Gneiting, T. (2002). Compactly supported correlation functions. *Journal of Multivariate Analysis*, 83:493–508.

Lawrence, N. (2003). Fast sparse Gaussian process methods: the informative vector machine. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems*. MIT Press.

Niessner, H. and Reichert, K. (1983). On computing the inverse of a sparse matrix. *International Journal for Numerical methods in Engineering*, 19:1513–1526.

Paciorek, C. and Schervish, M. (2006). Spatial modelling using a new class of nonstationary covariance functions. *Environmetrics*, 17:483–506.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal Of Machine Learning Research*, 6(3):1939–1959.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Rue, H. and Martino, S. (2007). Approximate Bayesian inference for hierarchical Gaussian Markov random field models. *Journal of Statistical Planning and Inference*, 137:3177–3192.

Seeger, M., Williams, C. K. I., and Lawrence, N. (2003). Fast forward selection to speed up sparse Gaussian process regression. In Bishop, C. M. and Frey, B. J., editors, *Ninth International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics.

Snelson, E. (2007). *Flexible and efficient Gaussian process models for machine learning*. PhD thesis, University College London.

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian process using pseudo-inputs. In Weiss, Y., Schlkopf, B., and Platt, J., editors, *Advances in Neural Information Processing Systems*, volume 18. The MIT Press.

Snelson, E. and Ghahramani, Z. (2007). Local and global sparse Gaussian process approximations. *Artificial Intelligence and Statistics*, 11.

Storkey, A. (1999). *Efficient Covariance Matrix Methods for Bayesian Gaussian Processes and Hopfield Neural Networks*. PhD thesis, University of London.

Takahashi, K., Fagan, J., and Chen, M.-S. (1973). Formation of a sparse bus impedance matrix and its application to short circuit study. In *Power Industry Computer Application Conference Proceedings*. IEEE Power Engineering Society.

Wendland, H. (2005). *Scattered Data Approximation*. Cambridge University Press.

Williams, C. K. I. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13. The MIT Press.

Continuous Time Dynamic Topic Models

Chong Wang
Computer Science Dept.
Princeton University
Princeton, NJ 08540

David Blei
Computer Science Dept.
Princeton University
Princeton, NJ 08540

David Heckerman
Microsoft Research
One Microsoft Way
Redmond, WA 98052

Abstract

In this paper, we develop the continuous time dynamic topic model (cDTM). The cDTM is a dynamic topic model that uses Brownian motion to model the latent topics through a sequential collection of documents, where a "topic" is a pattern of word use that we expect to evolve over the course of the collection. We derive an efficient variational approximate inference algorithm that takes advantage of the sparsity of observations in text, a property that lets us easily handle many time points. In contrast to the cDTM, the original discrete-time dynamic topic model (dDTM) requires that time be discretized. Moreover, the complexity of variational inference for the dDTM grows quickly as time granularity increases, a drawback which limits fine-grained discretization. We demonstrate the cDTM on two news corpora, reporting both predictive perplexity and the novel task of time stamp prediction.

1 Introduction

Tools for analyzing and managing large collections of electronic documents are becoming increasingly important. In recent years, *topic models*, which are hierarchical Bayesian models of discrete data, have become a widely used approach for exploratory and predictive analysis of text. Topic models, such as latent Dirichlet allocation (LDA) and the more general discrete component analysis [3, 4], posit that a small number of distributions over words, called topics, can be used to explain the observed collection. LDA is a probabilistic extension of latent semantic indexing (LSI) [5] and probabilistic latent semantic indexing (pLSI) [11]. Owing to its formal generative semantics, LDA has been extended and applied to authorship [19], email [15], computer vision [7], bioinformatics [18], and information retrieval [24]. For a good review, see [8].

Most topic models assume the documents are exchangeable in the collection, i.e., that their probability is invariant to permutation. Many document collections, such as news or scientific journals, evolve over time. In this paper, we develop the *continuous time dynamic topic model* (cDTM), which is an extension of the discrete dynamic topic model (dDTM) [2]. Given a sequence of documents, we infer the latent topics and how they change through the course of the collection.

The dDTM uses a state space model on the natural parameters of the multinomial distributions that represent the topics. This requires that time be discretized into several periods, and within each period LDA is used to model its documents. In [2], the authors analyze the journal *Science* from 1880-2002, assuming that articles are exchangeable within each year. While the dDTM is a powerful model, the choice of discretization affects the memory requirements and computational complexity of posterior inference. This largely determines the resolution at which to fit the model.

To resolve the problem of discretization, we consider time to be continuous. The continuous time dynamic topic model (cDTM) proposed here replaces the discrete state space model of the dDTM with its continuous generalization, Brownian motion [14]. The cDTM generalizes the dDTM in that the only discretization it models is the resolution at which the time stamps of the documents are measured.

The cDTM model will, generally, introduce many more latent variables than the dDTM. However, this seemingly more complicated model is simpler and more efficient to fit. As we will see below, from this formulation the variational posterior inference procedure can take advantage of the natural sparsity of text, the fact that not all vocabulary words are used at each measured time step. In fact, as the resolution gets finer, fewer and fewer words are used.

This provides an inferential speed-up that makes it possible to fit models at varying granularities. As examples, journal articles might be exchangeable within an issue, an assumption which is more realistic than one where they are exchangeable by year. Other data, such as news, might experience periods of time without any observation. While the dDTM requires representing all topics for the discrete ticks within these periods, the cDTM can analyze such data without a sacrifice of memory or speed. With the cDTM, the granularity can be chosen to maximize model fitness rather than to limit computational complexity.

We note that the cDTM and dDTM are not the only topic models to take time into consideration. Topics over time models (TOT) [23] and dynamic mixture models (DMM) [25] also include timestamps in the analysis of documents. The TOT model treats the time stamps as observations of the latent topics, while DMM assumes that the topic mixture proportions of each document is dependent on previous topic mixture proportions. In both TOT and DMM, the topics themselves are *constant*, and the time information is used to better discover them. In the setting here, we are interested in inferring evolving topics.

The rest of the paper is organized as follows. In section 2 we describe the dDTM and develop the cDTM in detail. Section 3 presents an efficient posterior inference algorithm for the cDTM based on sparse variational methods. In section 4, we present experimental results on two news corpora.

2 Continuous time dynamic topic models

In a time stamped document collection, we would like to model its latent topics as changing through the course of the collection. In news data, for example, a single topic will change as the stories associated with it develop. The discrete-time dynamic topic model (dDTM) builds on the exchangeable topic model to provide such machinery [2]. In the dDTM, documents are divided into sequential groups, and the topics of each slice evolve from the topics of the previous slice. Documents in a group are assumed exchangeable.

More specifically, a topic is represented as a distribution over the fixed vocabulary of the collection. The dDTM assumes that a discrete-time state space model governs the evolution of the natural parameters of the multinomial distributions that represent the topics. (Recall that the natural parameters of the multinomial are the logs of the probabilities of each item.) This is a time-series extension to the logistic normal distribution [26].

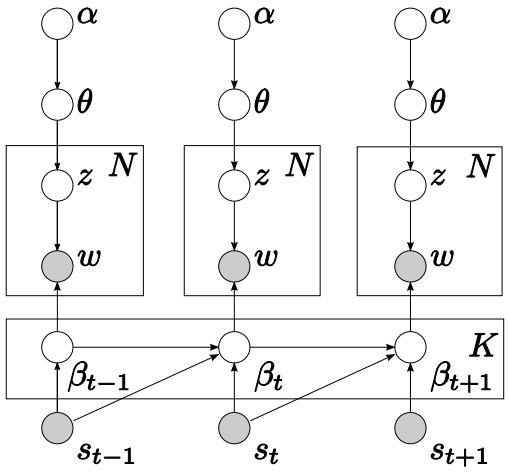


Figure 1: Graphical model representation of the cDTM. The evolution of the topic parameters β_t is governed by Brownian motion. The variable s_t is the observed time stamp of document d_t.

A drawback of the dDTM is that time is discretized. If the resolution is chosen to be too coarse, then the assumption that documents within a time step are exchangeable will not be true. If the resolution is too fine, then the number of variational parameters will explode as more time points are added. Choosing the discretization should be a decision based on assumptions about the data. However, the computational concerns might prevent analysis at the appropriate time scale.

Thus, we develop the continuous time dynamic topic model (cDTM) for modeling sequential time-series data with arbitrary granularity. The cDTM can be seen as a natural limit of the dDTM at its finest possible resolution, the resolution at which the document time stamps are measured.

In the cDTM, we still represent topics in their natural parameterization, but we use Brownian motion [14] to model their evolution through time. Let i, j $(j > i > 0)$ be two arbitrary time *indexes*, s_i and s_j be the time stamps, and Δ_{s_j,s_i} be the elapsed time between them. In a K-topic cDTM model, the distribution of the k^{th} $(1 \leq k \leq K)$ topic's parameter at term w is:

$$\begin{aligned} \beta_{0,k,w} &\sim \mathcal{N}(m, v_0) \\ \beta_{j,k,w}|\beta_{i,k,w}, s &\sim \mathcal{N}\left(\beta_{i,k,w}, v\Delta_{s_j,s_i}\right), \end{aligned} \quad (1)$$

where the variance increases linearly with the lag.

This construction is used as a component in the full generative process. (Note: if $j = i+1$, we write Δ_{s_j,s_i} as Δ_{s_j} for short.)

1. For each topic $k, 1 \leq k \leq K$,
   (a) Draw $\beta_{0,k} \sim \mathcal{N}(m, v_0 I)$.

2. For document d_t at time s_t ($t > 0$):
   (a) For each topic $k, 1 \leq k \leq K$,
      i. From the Brownian motion model, draw $\beta_{t,k}|\beta_{t-1,k}, s \sim \mathcal{N}(\beta_{t-1,k}, v\Delta_{s_t} I)$.
   (b) Draw $\theta_t \sim \text{Dir}(\alpha)$.
   (c) For each word,
      i. Draw $z_{t,n} \sim \text{Mult}(\theta_t)$.
      ii. Draw $w_{t,n} \sim \text{Mult}(\pi(\beta_{t,z_{t,n}}))$.

The function π maps the multinomial natural parameters, which are unconstrained, to its mean parameters, which are on the simplex,

$$\pi(\beta_{t,k})_w = \frac{\exp(\beta_{t,k,w})}{\sum_w \exp(\beta_{t,k,w})}. \quad (2)$$

The cDTM is illustrated in Figure 1.

The cDTM can be seen as a generalization of the dDTM. Both models assume that the log probability of a term exhibits variance over an interval of time between observations. In the dDTM, this interval is evenly divided into discrete ticks. A parameter controls the variance at each tick, and the variance across the whole interval is that parameter multiplied by the number of ticks. As a consequence of this representation, the topic, i.e., the full distribution over terms, is explicitly represented at each tick. For fine-grained time series, this leads to high memory requirements for posterior inference, even if the observations are sparsely distributed throughout the timeline.

In the cDTM, however, the variance is a function of the lag between observations, and the probabilities at discrete steps between those observations need not be considered. Inference, as we will see below, can be handled sparsely. Thus, choosing the right granularity becomes a modeling issue rather than one governed by computational concerns. A dDTM is obtained with a cDTM by measuring the time stamps of the documents at the desired granularity.

Akin to Brownian motion as the limiting process of a discrete-time Gaussian random walk [6], the cDTM is the limiting process of the dDTM. Denote the per-tick variance in the dDTM by σ^2, and note that it is a function of the tick granularity (to make models comparable). The cDTM is the limiting model in this setting as σ^2 approaches zero. We emphasize that with the cDTM, we need not represent the log probabilities at the ticks between observed documents. This perspective is illustrated in Figure 2.

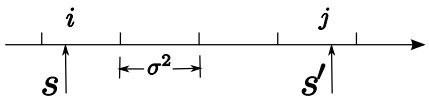


Figure 2: Documents are available only at time s and s', and no documents between them. When $\sigma^2 \to 0$, the dDTM becomes a cDTM, and we no longer need to represent the steps between i and j.

3 Sparse variational inference

The central problem in topic modeling is posterior inference, i.e., determining the distribution of the latent topic structure conditioned on the observed documents. In sequential topic models, this structure comprises the per-document topic proportions θ_d, per-word topic assignments $z_{d,n}$, and the K sequences of topic distributions $\beta_{t,k}$. The true posterior is not tractable [2]. We must appeal to an approximation.

Several approximate inference methods have been developed for topic models. The most widely used are variational inference [3, 20] and collapsed Gibbs sampling [9]. In the sequential setting collapsed Gibbs sampling is not an option because the distribution of words for each topic is not conjugate to the word probabilities. Thus, we employed variational methods.

The main idea behind variational methods is to posit a simple family of distributions over the latent variables, indexed by free *variational parameters*, and to find the member of that family which is closest in Kullback-Leibler divergence to the true posterior. Good overviews of this methodology can be found in [12] and [22]. For continuous time processes, variational inference has been applied in Markov jump processes [1] and diffusion processes [17], where the variational distributions are also random processes.

For the cDTM described above, we adapt variational Kalman filtering [2] to the continuous time setting. For simplicity, assume that one document occurs at each time point. In their algorithm, the variational distribution over the latent variables is:

$$q(\beta_{1:T}, z_{1:T,1:N}, \theta_{1:T} \mid \hat{\beta}, \phi, \gamma) = \prod_{k=1}^{K} q(\beta_{1,k}, \ldots, \beta_{T,k} | \hat{\beta}_{1,k}, \ldots, \hat{\beta}_{T,k}) \times \prod_{t=1}^{T} \left(q(\theta_t|\gamma_t) \prod_{n=1}^{N_t} q(z_{t,n}|\phi_{t,n}) \right). \quad (3)$$

The variational parameters are a Dirichlet γ_t for the per-document topic proportions, multinomials ϕ for each word's topic assignment, and $\hat{\beta}$ variables, which are "observations" to a variational Kalman filter.

These variables are fit such that the approximate posterior is close to the true posterior. From the variational Kalman filter, the $\beta_{k,t}, 1 \leq t \leq T$ retain their chained structure in the variational distribution. Vari-

ational inference proceeds by coordinate ascent, updating each of these parameters to minimize the KL between the true posterior and variational posterior.

For simplicity, now we consider a model with only *one* topic. These calculations are simpler versions of those we need for the more general latent variable model but exhibit the essential features of the algorithm. For the cDTM, we assume a similar variational distribution, with the same variational Dirichlet and variational multinomials for the per-document variables. The cDTM updates for these parameters are identical to those in [2], and we do not replicate them here.

In principle, we can directly use the variational Kalman filtering algorithm for the cDTM by replacing the state space model with Brownian motion. Let $\mathcal{V}$ be the size of the vocabulary. While conceptually straightforward, this will yield $\mathcal{V}T$ variational parameters in the vectors $\hat{\beta}_{1:T}$. When T and $\mathcal{V}$ are large, as in a fine-graned model, posterior inference will require massive amounts of time and memory. Thus, we develop a sparse variational inference procedure, which significantly improves its complexity without sacrificing accuracy.

The main idea behind the sparse variational Kalman filtering algorithm is that if certain $\beta_{t,w}$ do not describe any term emissions, i.e., there are no observations of w at t, then the true posterior of $\beta_{t,w}$ is only determined by the observations of the other words at that time. Therefore, we don't need to explicitly represent $\hat{\beta}_{t,w}$ for those w that are not observed.

Figure 3 illustrates the idea behind sparse variational inference for the cDTM. In Figure 3, the variational posterior of the log probability of a word $\beta_{t,w}$ is determined by the variational observations of the observed words. From the belief propagation point of view, the belief propagated from $\beta_{t,w}$ to node $\beta_{t+2,w}$ is not revised by term w, and this property is retained in the sparse variational inference algorithm. The probability of variational observation $\hat{\beta}_{t,w}$ given $\beta_{t,w}$ is a Gaussian:

$$\hat{\beta}_{t,w}|\beta_{t,w} \sim \mathcal{N}(\beta_{t,w}, \hat{v}_t). \tag{4}$$

We next describe the forward-backward algorithm for the sparse variational Kalman filter, which is needed to compute the expectations for updating the variational parameters. For a certain term w, the variational forward distribution $p(\beta_{t,w}|\hat{\beta}_{i,i\leq t,w})$ is a Gaussian [13] and can be characterized as follows.

$$\begin{aligned}
\beta_{t,w}|\hat{\beta}_{i,i\leq t,w} &\sim \mathcal{N}(m_{t,w}, V_{t,w}) \\
m_{t,w} &= \mathbb{E}(\beta_{t,w}|\hat{\beta}_{i,i\leq t,w}) \\
V_{t,w} &= \mathbb{E}((\beta_{t,w} - m_{t,w})^2|\hat{\beta}_{i,i\leq t,w}).
\end{aligned} \tag{5}$$

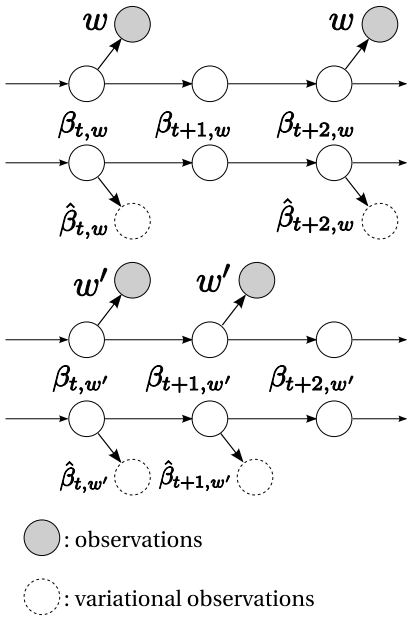


Figure 3: A simplified graphical model shows how sparse variational inference works with only single topic. Note this generation process needs normalization to β_t according to Equation 2, but this will not affect the sparse solution. For term w, there are no observations at time index $t+1$ (or time s_{t+1}), the corresponding variational observations don't appear at time index $t+1$. For term w', there are no observations at time index $t+2$ (or time s_{t+2}), the corresponding variational observations don't appear at time index $t+2$.

If w is not observed at time step t then

$$\begin{aligned}
m_{t,w} &= m_{t-1,w} \\
V_{t,w} &= P_{t,w}, \\
P_{t,w} &= V_{t-1,w} + v\Delta_{s_t},
\end{aligned} \tag{6}$$

which means that the forward mean remains the same as the previous step. Otherwise,

$$\begin{aligned}
m_{t,w} &= \frac{\hat{\beta}_{t,w}P_{t,w} + \hat{v}_t m_{t-1,w}}{P_{t,w} + \hat{v}_t} \\
V_{t,w} &= \hat{v}_t \frac{P_{t,w}}{P_{t,w} + \hat{v}_t} \\
\hat{\beta}_{t,w}|\hat{\beta}_{i,i\leq t-1,w} &\sim \mathcal{N}(m_{t-1,w}, P_{t,w} + \hat{v}_t).
\end{aligned} \tag{7}$$

Similarly, the variational backward distribution $p(\beta_{t,w}|\hat{\beta}_{i,i\leq T,w})$ is also a Gaussian:

$$\begin{aligned}
\beta_{t,w}|\hat{\beta}_{i,i\leq T,w} &\sim \mathcal{N}(\widetilde{m}_{t,w}, \widetilde{V}_{t,w}) \\
\widetilde{m}_{t,w} &= \mathbb{E}(\beta_{t,w}|\hat{\beta}_{i,i\leq T,w}) \\
\widetilde{V}_{t,w} &= \mathbb{E}((\beta_{t,w} - \widetilde{m}_{t,w})^2|\hat{\beta}_{i,i\leq T,w}). \\
\widetilde{m}_{t-1,w} &= m_{t-1,w}\frac{f_t v}{P_{t,w}} + \widetilde{m}_{t,w}\frac{V_{t-1,w}}{P_{t,w}}
\end{aligned}$$

$$\widetilde{V}_{t-1,w} = V_{t-1,w} + \frac{V_{t-1,w}^2}{P_{t,w}^2}(\widetilde{V}_{t,w} - P_{t,w}). \quad (8)$$

With this forward-backward computation in hand, we turn to optimizing the variational observations $\hat{\beta}_{w,k}$ in the sparse setting. Equivalent to minimizing KL is tightening the bound on the likelihood of the observations given by Jensen's inequality [12].

$$\mathcal{L}(\hat{\beta}) \geq \sum_{t=1}^{T} \mathbb{E}_q \left[(\log p(\mathbf{w}_t|\beta_t) + \log p(\beta_t|\beta_{t-1})\right] + H(q), \quad (9)$$

where $H(q)$ is the entropy. This is simplified to

$$\begin{aligned}\mathcal{L}(\hat{\beta}) &\geq \sum_{t=1}^{T} \mathbb{E}_q \left[\log p(\mathbf{w}_t|\beta_t) - \log q(\hat{\beta}_t|\beta_t)\right] \\ &\quad + \sum_{t=1}^{T} \log q(\hat{\beta}_t|\hat{\beta}_{i,i\leq t-1}), \quad (10)\end{aligned}$$

We use $\delta_{t,w} = 1$ or 0 to represent whether $\hat{\beta}_{t,w}$ is in the variational observations or not. Then the terms above are

$$\begin{aligned}\mathbb{E}_q \log q(\mathbf{w}_t|\beta_t) &\geq \sum_w n_{t,w}\widetilde{m}_{t,w} \\ &\quad -n_t \log \sum_w \exp(\widetilde{m}_{t,w} + \widetilde{V}_{t,w}/2) \\ \mathbb{E}_q \log p(\hat{\beta}_t|\beta_t) &= \sum_w \delta_{t,w}\mathbb{E}_q \log q(\hat{\beta}_{t,w}|\beta_{t,w}) \\ \log q(\hat{\beta}_t|\hat{\beta}_{i,i\leq t-1}) &= \sum_w \delta_{t,w} \log q(\hat{\beta}_{t,w}|\hat{\beta}_{i,i\leq t-1,w}).\end{aligned}$$

The count of w in document d_t is $n_{t,w}$ and $n_t = \sum_w n_{t,w}$.

Thus, to optimize the variational observations, we need only to compute the derivative $\partial\mathcal{L}/\partial\hat{\beta}_{t,w}$ for those $\delta_{t,w} = 1$. The general memory requirement is $\mathcal{O}(\sum_t \sum_w \delta_{t,w})$—the sum of the number of unique terms at each time point—which is usually much smaller than $\mathcal{O}(\mathcal{V}T)$, the memory requirement for the densely represented algorithm. Formally, we can define the sparsity of the data set to be

$$\text{sparsity} = 1 - \left(\sum_t \sum_w \delta_{t,w}\right)/(\mathcal{V}T), \quad (11)$$

which we will compute for several data sets in the next section. Finally, we note that we use the conjugate gradient algorithm [16] to optimize the variational observations from these partial derivatives.

As an example of the speed-up offered by sparse variational inference, consider the *Science* corpus from 1880-2002, analyzed by [2], which contains 6243 issues of the magazine. Note that these issues are not evenly spaced over the time line. In the dDTM, these documents were separated by years. To analyze them at a finer scale, e.g., issue by issue, one needs to consider 6243 time points. With a vocabulary size of 5000, for a 10-topic setting, the cDTM requires 0.8G memory while the dDTM requires 2.3G memory, nearly 3 times larger. The sparsity of *Science* is 0.65. This means that a term only appears in about a third of the total time points.

	Sparsity			
Data set	Hour	Day	Week	Month
AP	0.93	0.68	0.12	–
Election 08	–	0.95	0.79	0.50

Table 1: Sparsity for two data sets where available. Higher numbers indicate a sparser data set and more efficiency for the cDTM over the dDTM.

4 Experiments

In this section, we demonstrate the cDTM model on two news corpora. We report predictive perplexity and a results on the novel task of time stamp prediction.

4.1 News Corpora

We used two news corpora. First, "AP" is a subset from the TREC AP corpus [10] containing the news from 05/01/1988 to 06/30/1988. We extracted the documents about the presidential election in 1988 resulting in 1,342 documents. These documents are time stamped by hour. Second, the "Election 08" data are summaries of the top articles from Digg[1] classified as being part of the 2008 presidential election. We used articles from 02-27-2007 to 02-22-2008. This data set has 1,040 summaries. Time is measured in days.

Table 1 shows the sparsity information for these data in terms of the resolution at which we can analyze them. This illustrates the gain in efficiency of the cDTM. For example, in the day setting of the Election 08 data, the sparsity is 0.95. The dDTM model will need at least 20 times more parameters than the cDTM to analyze the data at this resolution.

4.2 Per-Word Predictive Perplexity

Let D_t be the set of documents at time index t. We performed approximate posterior inference on these data with the cDTM at different levels of granularity. To make models comparable, we set the variance across the entire period to be the same (see Equation 1). We evaluated the models with perplexity. Specifi-

[1] http://digg.com

cally, we computed the per-word predictive perplexity of the documents at time t based on the data of the previous $t-1$ time indices,

$$\text{perplexity}_{\text{pw}}(t) = \exp\left\{-\frac{1}{|D_t|}\sum_{d\in D_t}\frac{\log p(\mathbf{w}_d|D_{1:t-1})}{N_d}\right\}. \quad (12)$$

Note that lower numbers are better.

Since each document is predicted exactly once in all models at different granularities, we also compute the *averaged* per-word perplexity over the time line, which is defined as

$$\text{perplexity}_{\text{pw}} = \exp\left\{-\frac{\sum_{d\in D}\log p(\mathbf{w}_d)}{\sum_{d\in D} N_d}\right\}. \quad (13)$$

In the AP data, we made predictions from 5/15/1988 to 05/29/1988. In the Election 08 data, we made predictions from 04/26/2007 to 02/22/2008. Figure 4 shows the results of the per-word predictive perplexity over the time line on both data sets for the 10 topic model. Figure 5 shows the results of average per-word perplexity for 1, 3, 5 and 10 topics.

From the computational perspective, we note that the sparse inference algorithm lets us fit models of different granularities efficiently. For the AP data, the day model and week are almost comparable. Models with 5 and 10 topics perform better.

In the Election 08 data, the 1-topic model performs best. We suspect that this is because the summaries are very short. More complex models, i.e., those with more topics, are not appropriate. The models perform differently at different levels of granularity because the amount of data supported at each time point depends on the chosen level. It is not necessarily the case that a finer grained model will contain enough data to provide a better predictive distribution.

4.3 Time Stamp Prediction

We can further use the cDTM for *time stamp prediction*, dating a document based on its content. To assess this task, we split each data set into 80% training and 20% testing sets. We predict the time stamp of each test document by finding its most likely location over the time line. We measure the error in terms of the same granularity at which the data are measured.

We investigated two approaches. The first is the *flat* approach. Each model of different granularity predicts as best it can. The second is the *hierarchical* approach. We use models of increasing granularity to "zoom in" on the prediction. For example, to predict the day, we first find the best month, then the best week within the month, and then the best day within the week. We compute the average absolute error over the test data set. Figure 6 illustrates the results.

The hierarchical approach always performs better than or as well as the flat approach. The hour model in the AP data and day model in Election 08 perform worse. With the small data sets, a larger granularity is better. The reason may also lie in the parameter v. Currently it is shared among all models. In the future, we'd like to infer it from the data.

4.4 Example Topics

We provide some example topics by using the week model in the Election 08 data. We sample the topics every two months. Figure 7 shows one of the topics. At the beginning the election (year 2007), general issues were discussed more, such as "healthcare." As the competition went up (year 2008), the topics were more about candidates themselves and changing faster.

5 Conclusions

In this paper, we have developed the cDTM, using Brownian motion to model continuous-time topic evolution. The main advantage of the cDTM is that we can employ sparse variational inference for fast model comparison. We demonstrated the use of cDTM by measuring the predictive likelihood and time stamp prediction accuracy on two real-world data sets. In future work, we plan to explore the Ornstein-Uhlenbeck (OU) model [21], a generalization of Brownian model, that allows bounded variance.

Acknowledgments. We thank anonymous reviewers for their valuable comments. We would also like to thank Jordan Boyd-Graber and Jonathan Chang for many insightful discussions. David M. Blei is supported by grants from Microsoft Research, Google, and the Office of Naval Research (175-6343).

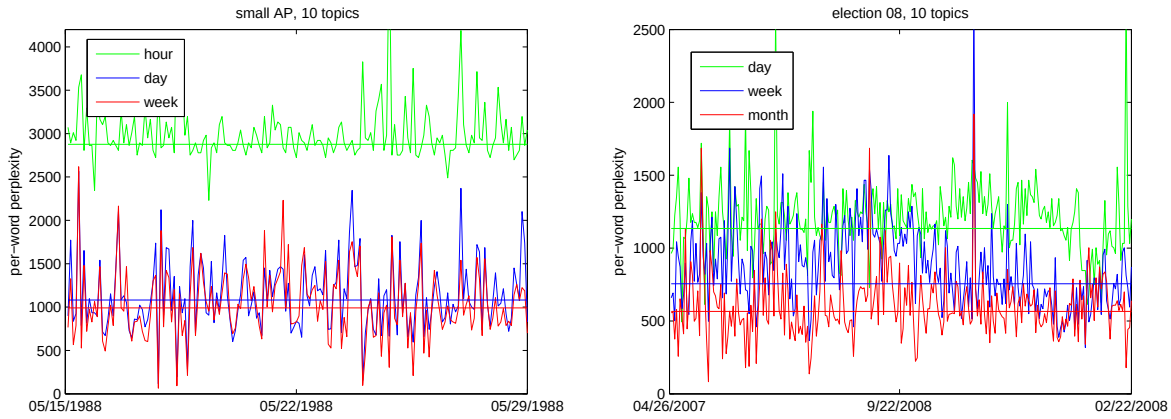
References

[1] C. Archambeau, M. Opper, Y. Shen, D. Cornford, and J. Shawe-Taylor. Variational inference for diffusion processes. In *NIPS*, 2007.

[2] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *ICML*, 2006.

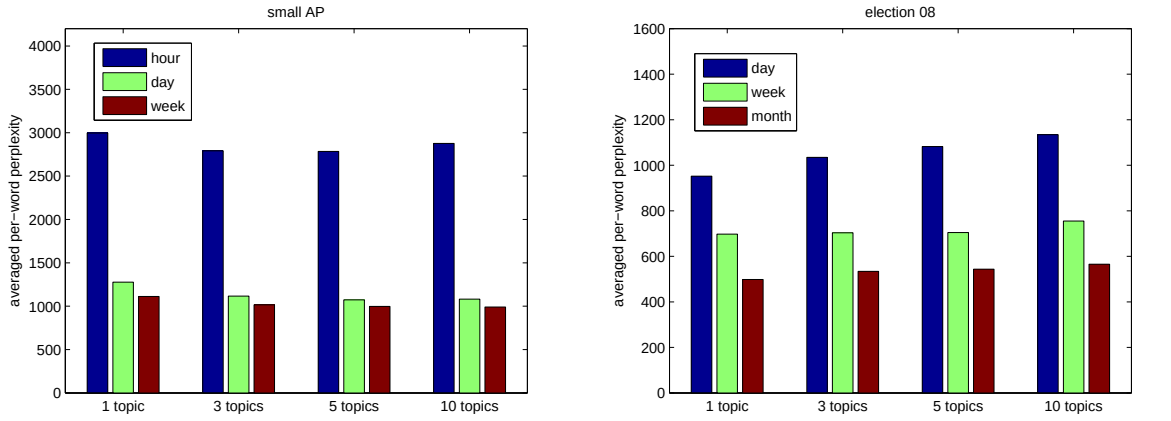
[3] D. M. Blei, A. Ng, and M. I. Jordan. Latent Dirichlet allocation. *JMLR*, 3:993–1002, 2003.

[4] W. Buntine and A. Jakulin. Applying discrete PCA in data analysis. In *UAI*, 2004.



(a) (b)

Figure 4: Per-word predictive perplexity comparison. The straight lines are the corresponding averaged per-word predictive perplexities. (a) AP data. The week model performs the best, but the day model is almost comparable. (b) Election 08 data. The month model performs the best.



(a) (b)

Figure 5: Averaged per-word predictive perplexity comparison. (a) AP data. The week model performs the best, but the day model is almost comparable. (b) Election 08 data. The month model performs the best.

[5] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[6] J. Durbin and S. Koopman. *Time Series Analysis by State Space Methods*. Oxford Univ. Press, 2001.

[7] L. Fei-Fei and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005.

[8] T. Griffiths and M. Steyvers. Probabilistic topic models. In *Latent Semantic Analysis: A Road to Meaning*. 2006.

[9] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proc. Natl. Acad. Sci.*, 101 Suppl 1:5228–5235, April 2004.

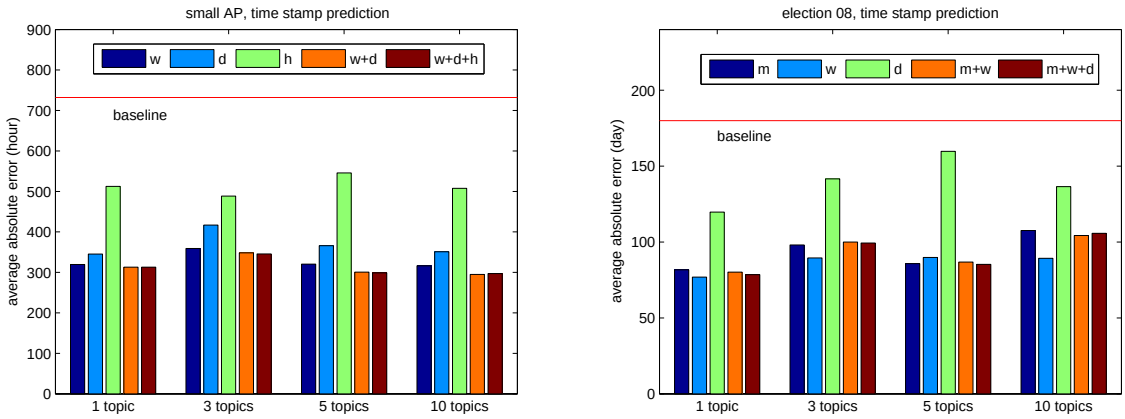
[10] D. Harman. Overview of the first text retrieval conference (TREC-1). In *TREC-1*, 1992.

[11] T. Hofmann. Probabilistic latent semantic analysis. In *UAI*, 1999.

[12] M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[13] R. Kalman. A new approach to linear filtering and prediction problems. *Transaction of the AMSE: Journal of Basic Engineering*, 82:35–45, 1960.

[14] G. F. Lawler. *Introduction to Stochastic Processes*. Chapman & Hall/CRC, 1995.



(a) (b)

Figure 6: Time stamp prediction. 'm' stands for *flat* approach of 'month', 'w' for 'week', 'd' for 'day' and and 'h' for 'hour.' 'm+w' stands for the *hierarchical* approach of combining 'month' and 'week', and 'm+w+d', 'w+d', 'w+d+h' are similarly defined. The baseline is the expectation of the error by randomly assigning a time. (a) AP data. 5-topic and 10-topic models perform better than others, and the hierarchical approach always achieves the best performance. (b) Election 08 data. 1-topic model performs best due to the short documents. The hierarchical approach achieves comparable performances.

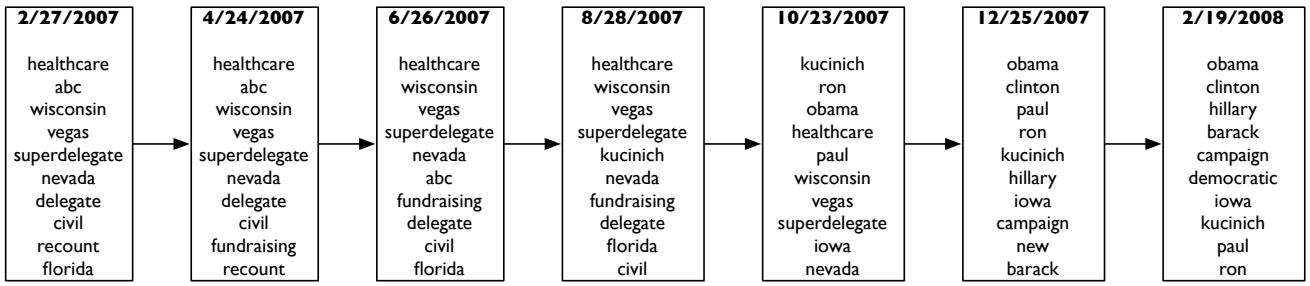


Figure 7: Examples from a 3-topic cDTM using the week model in the Election 08 data. In year 2007, the topics were more about general issues, while around year 2008, were more about candidates and changing faster.

[15] A. McCallum, A. Corrada-Emmanuel, and X. Wang. Topic and role discovery in social networks. In *IJCAI*, 2005.

[16] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.

[17] M. Opper and G. Sanguinetti. Variational inference for markov jump processes. In *NIPS*, 2007.

[18] S. Rogers, M. Girolami, C. Campbell, and R. Breitling. The latent process decomposition of cDNA microarray data sets. *IEEE/ACM Trans. on Comp. Bio. and Bioinf.*, 2(2):143–156, 2005.

[19] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *UAI*, 2004.

[20] Y. W. Teh, D. Newman, and M. Welling. A collapsed variational bayesian inference algorithm for latent Dirichlet allocation. In *NIPS*, 2006.

[21] G. E. Uhlenbeck and L. S. Ornstein. On the theory of Brownian motion. *Phys. Rev.*, 36:823–41, 1930.

[22] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families and variational inference. Technical Report 649, UC Berkeley, Dept. of Statistics, 2003.

[23] X. Wang and A. McCallum. Topics over time: A non-Markov continuous-time model of topical trends. In *SIGKDD*, 2006.

[24] X. Wei and B. Croft. LDA-based document models for ad-hoc retrieval. In *SIGIR*, 2006.

[25] X. Wei, J. Sun, and X. Wang. Dynamic mixture models for multiple time series. In *IJCAI*, 2007.

[26] M. West and J. Harrison. *Bayesian Forecasting and Dynamic Models*. Springer, 1997.

Hybrid Variational/Gibbs Collapsed Inference in Topic Models

Max Welling*
Dept. Computer Science
University of California Irvine
Irvine, CA, USA
welling@ics.uci.edu

Yee Whye Teh
Gatsby Computational Neuroscience Unit
University College London
London, UK
ywteh@gatsby.ucl.ac.uk

Bert Kappen
Dept. Biophysics
Radboud University Nijmegen
Nijmegen, The Netherlands
b.kappen@science.ru.nl

Abstract

Variational Bayesian inference and (collapsed) Gibbs sampling are the two important classes of inference algorithms for Bayesian networks. Both have their advantages and disadvantages: collapsed Gibbs sampling is unbiased but is also inefficient for large count values and requires averaging over many samples to reduce variance. On the other hand, variational Bayesian inference is efficient and accurate for large count values but suffers from bias for small counts. We propose a hybrid algorithm that combines the best of both worlds: it samples very small counts and applies variational updates to large counts. This hybridization is shown to significantly improve test-set perplexity relative to variational inference at no computational cost.

1 Introduction

Bayesian networks (BNs) represent an important modeling tool in the field of artificial intelligence and machine learning (Heckerman, 1999). In particular the subclass of BNs known as "topic models" is receiving increasing attention due to its success in modeling text as a bag-of-words and images as a bag-of-features (Blei et al., 2003). Unlike most applications of Bayesian networks, we will be interested in "Bayesian Bayesian networks" where we also treat the conditional probability tables (CPTs) as random variables (RVs). The key computational challenge for these models is inference, namely estimating the posterior distribution over both parameters and hidden variables, and ultimately estimating predictive probabilities and the marginal log-likelihood (or evidence).

It has been argued theoretically (Castella & Robert, 1996) and observed empirically in topic models (Griffiths & Steyvers, 2002; Buntine, 2002) that Gibbs sampling in a collapsed state space where the CPTs have been marginalized out leads to efficient inference. It is expected that this also holds more generally true for discrete Bayesian networks. Variational Bayesian (VB) approximations have also been applied to topic models (Blei et al., 2003) but predictive probability results have consistently been inferior to collapsed Gibbs sampling (CGS). More recently, variational approximations have been extended to operate in the same collapsed state space of CGS (Teh et al., 2006; Teh et al., 2008). These collapsed variational Bayesian (CVB) inference algorithms improve upon VB but still lag behind CGS.

In this paper we will propose a hybrid inference scheme that combines CGS with VB approximations. The idea is to split all data-cases into two sets, the ones that will be treated variationally and the ones that will be treated through sampling. The two approximations interact in a consistent manner in the sense that both VB and CGS updates are derived from a single objective function. The advantage of the VB updates is that they scale better computationally. We show empirically that hybrid algorithms achieve almost the same accuracy as CGS because they are only applied where they are expected to work well. The algorithm can be seen to trade off bias with variance in a flexible and tunable manner.

2 Topic Models as Bayesian Networks

We will assume that all visible and hidden variables are discrete. However, we expect the results to hold more generally for models in the exponentially family. We will first develop the theory for latent Dirichlet allocation (LDA) (Blei et al., 2003) and later generalize the results to Bayesian networks. To facilitate the transition from LDA to BNs we will treat LDA in a slightly unconventional way by using a single index i that runs over all words in all documents (in contrast to the index ij which is conventional for LDA), see Fig.1. LDA is equivalent to the Bayesian network $d_i \rightarrow z_i \rightarrow x_i$, where nodes $x_i = w$ (word-type) and $d_i = j$

*On Sabbatical at Radboud University, Netherlands, Department of Biophysics.

(document label) have been observed. The topic variable is indicated by $z_i = k$. In the following we will also use the indicator variables $X_{iw} = \mathbb{I}[x_i = w]$, $D_{ij} = \mathbb{I}[d_i = j]$ and $Z_{ik} = \mathbb{I}[z_i = k]$.

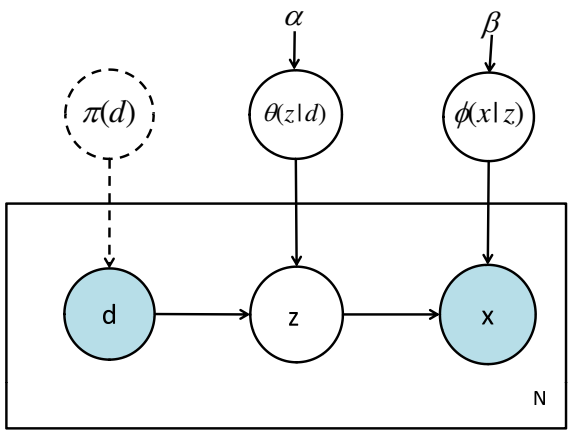


Figure 1: LDA depicted as a standard Bayesian network. N runs over all word-tokens and for each token, both the word-type x and the document label j are observed. The topic variable z is hidden.

The joint distribution is given as the product of the probability of each variable given its parents, $p(x_i = w|z_i = k) = \phi_{wk}$, $p(z_i = k|d_i = j) = \theta_{kj}$ and $p(d_i = j) = \pi_j$. The last term decouples from the rest of the model because d_i is always observed. For each of these CPTs we place a Dirichlet prior, $\phi_{:,k} \sim \mathcal{D}(\beta)$ and $\theta_{:,j} \sim \mathcal{D}(\alpha_:)$. Note that we have chosen a scalar strength parameter β (i.e. a symmetric Dirichlet prior) for ϕ but vector valued strength parameters $\alpha_: = \{\alpha_k\}$ for θ.

Next we integrate out the CPTs. Since the Dirichlet priors are conjugate to the discrete conditional probabilities of the Bayesian network, this is possible analytically, resulting in the following expression,

$$p(\{z_i, x_i\}|\{d_i\}) \propto \frac{\prod_{wk} \Gamma(N_{wk}+\beta)}{\prod_k \Gamma(N_k+W\beta)} \prod_{jk} \Gamma(N_{kj}+\alpha_k) \quad (1)$$

where W is the total number of different word types (the vocabulary size), and the counts are defined as follows:

$$N_{wkj} = \sum_i X_{iw} Z_{ik} D_{ij}, \quad (2)$$

$N_{wk} = \sum_j N_{wkj}$, $N_{kj} = \sum_w N_{wkj}$ and $N_k = \sum_{wj} N_{wkj}$. Note that N_{wkj} are subject to the observational constraints $\sum_k N_{wkj} = \hat{N}_{wj}$ given by the training corpus.

3 Standard Variational Bayes

In the variational Bayesian framework we approximate the posterior distribution with a factorized one:

$$p(\{z_i, \phi, \theta\}|\{x_i, d_i\}) \approx \prod_k Q(\phi_{:,k}|\xi_{:,k}) \prod_j Q(\theta_{:,j}|\zeta_{:,j}) \prod_i Q(z_i) \quad (3)$$

Solving for $Q(\phi_{:,k})$ and $Q(\theta_{:,j})$, we find that they are Dirichlet as well, with parameters,

$$\xi_{wk} = \beta + \bar{N}_{wk}, \qquad \zeta_{kj} = \alpha_k + \bar{N}_{kj} \quad (4)$$

where $\bar{N}_{wk} = \sum_i Q_{ik} X_{iw}$, $\bar{N}_{kj} = \sum_i Q_{ik} D_{ij}$ and $Q_{ik} = Q(z_i = k)$. After plugging these updates back into the update for $Q(z_i)$ we find,

$$Q_{ik} \propto \frac{\exp(\psi(\bar{N}_{x_i k} + \beta))}{\exp(\psi(\bar{N}_k + W\beta))} \exp(\psi(\bar{N}_{kd_i} + \alpha_k)) \quad (5)$$

where $\psi(\cdot) = \frac{\partial}{\partial x} \log \Gamma(x)$ is the derivative of the log-gamma function.

For future reference we shall now derive the same result in a different manner to which we shall refer as "standard variational Bayes" (SVB) in the following. We first marginalize out the CPTs and consider a variational lower bound on the evidence with a variational distribution $Q(\{z_i\})$ (without assuming factorization). Lemma 1 in the appendix shows that the expression for the log-probability $\log p(\{z_i, x_i\}|\{d_i\})$ is a convex function of $\{N_{wjk}\}$. This allows us to move the average over $Q(\{z_i\})$ inside the expression for the log-probability, at the cost of introducing a further lower bound on the evidence. The resulting expression is now precisely the logarithm of Eqn.1 with the counts N_{wkj} replaced by average counts $\bar{N}_{wkj}$,

$$\bar{N}_{wkj} = \mathbb{E}[N_{wkj}]_Q = \hat{N}_{wj} Q_{k|wj} \quad (6)$$

in terms of which we define

$$\bar{N}_{wk} = \mathbb{E}[N_{wk}]_Q = \sum_j \hat{N}_{wj} Q_{k|wj} \quad (7)$$

$$\bar{N}_{kj} = \mathbb{E}[N_{kj}]_Q = \sum_w \hat{N}_{wj} Q_{k|wj} \quad (8)$$

$$\bar{N}_k = \mathbb{E}[N_k]_Q = \sum_{wj} \hat{N}_{wj} Q_{k|wj} \quad (9)$$

where $\hat{N}_{wj}$ are the observed word-document counts. To understand the definition of $Q_{k|wj}$ we note that $Q(z_i)$ for all i's with the same values of $x_i = w$, $d_i = j$ are equal, so without loss of generality we can use a single set of parameters $Q_{k|wj} = Q(z_i = k)$ for all data-cases i which share the same observed labels w, j.

The final step is to variationally bound this expression once more, using again the fact that the sum of log-gamma factors is a convex function,

$$F(\bar{N}^*_{wkj}) \geq F(\bar{N}_{wjk}) + \sum_{wkj} \nabla_{wkj} F(\bar{N}_{wkj})(\bar{N}^*_{wjk} - \bar{N}_{wkj}) \tag{10}$$

where $\bar{N}_{wkj}$ is held fixed. Recalling definition 6 and taking derivatives w.r.t $Q_{k|wj}$ gives the following update,

$$Q_{k|wj} \propto \frac{\exp(\psi(\bar{N}_{wk} + \beta))}{\exp(\psi(\bar{N}_k + W\beta))} \exp(\psi(\bar{N}_{jk} + \alpha_k)) \tag{11}$$

The factorization follows directly from the update and is a result of Jensen's inequality. We can alternatively arrive at Eqn.11 without assuming the second bound of Eqn.10 by assuming that $Q(\{z_i\})$ factorizes and equating its derivatives to 0. However, this does not guarantee convergence as $Q_{k|wj}$ now also appears on the RHS of Eqn.11. Note that Eqn.11 is equivalent but looks subtly different from Eqn.5 in that it avoids updating variational distributions for data-cases i with the same labels w, j. As a result it scales more favorably as the number of unique word-document pairs in the training corpus rather than the total number of word tokens.

4 Collapsed Gibbs Sampling

An alternative to variational inference is collapsed Gibbs sampling where one samples each z_i in turn, given the values of the remaining $z_{\neg i}$. The conditional probabilities are easily calculated from Eqn.1,

$$p(z_i = k | \mathbf{z}_{\neg i}, \mathbf{x}, \mathbf{d}) \propto \frac{(N^{\neg i}_{wk} + \beta)}{(N^{\neg i}_k + W\beta)} (N^{\neg i}_{jk} + \alpha_k) \tag{12}$$

where the superscript $^{\neg i}$ denotes that data-case i has been removed from the count and we have assumed that $x_i = w$ and $d_i = j$.

Given samples at equilibrium one can obtain unbiased estimates of quantities of interest. The trade-off is that one needs to average over many samples to reduce the effects of sampling noise. Computationally, CGS scales as $\mathcal{O}(NK)$ in time and $\mathcal{O}(N)$ in space, where N is the total number of words in the corpus and K is the number of topics. This in contrast to the SVB updates in the previous section for which both time and space scale as $\mathcal{O}(MK)$ with M the number of unique word-document pairs.

In the following section we will derive a principled hybridization of SVB and CGS that can be viewed as a tunable trade-off between bias, variance and computational efficiency.

5 The Hybrid SVB/CGS Algorithm

The high level justification for a hybrid algorithm is the intuition that the evidence only depends on count arrays, N_{wk}, N_{kj} and N_k, which are sums of assignment variables $N_{wkj} = \sum_i X_{iw} Z_{ik} D_{ij}$ where only Z is random. Also the Rao-Blackwellised estimate of the predictive distribution is a function of these counts,

$$p(x^* = w | \{x_i, z_i, d_i\}, d^* = j) = \sum_k \frac{N_{wk} + \beta}{N_k + W\beta} \frac{N_{kj} + \alpha_k}{N_j + \sum_k \alpha_k} \tag{13}$$

The central limit theorem tells us that sums of random variables tend to concentrate and behave like a normal distribution under certain conditions. Moreover, the variance/covariance of the predictive distribution is expected to scale with $1/n$, where n is the number of data-cases that contribute to the sum. We expect that variational approximations work well for large counts.

These insights make it natural to split the dataset into two subsets, one subset $\mathcal{S}^{\text{VB}}$ to which we will apply the VB approximation and the complement $\mathcal{S}^{\text{CG}}$ to which we shall apply collapsed Gibbs sampling. In practice we have chosen these sets to be:

$$\mathcal{S}^{\text{VB}} = \{i | \hat{N}_{x_i, d_i} > r\}, \qquad \mathcal{S}^{\text{CG}} = \{i | \hat{N}_{x_i, d_i} \leq r\} \tag{14}$$

In the experiments below we have chosen $r = 1$. Although we do not expect that central limit tendencies apply to counts smaller than about 10, we have chosen this extreme setting to convey an important conclusion, namely that the counts with value $\hat{N}_{wj} = 1$ already explain much of the difference between VB and CGS algorithms.

We will assume the following factorization for the variational posterior, $Q = Q^{\text{VB}} Q^{\text{CG}}$. Moreover, in the derivation below it will follow that Q^{VB} becomes factorized as well, i.e. $Q^{\text{VB}} = \prod_i Q^{\text{VB}}_i$.

The evidence of the collapsed distribution under these assumptions reads,

$$\mathcal{E} = \mathcal{H}(Q^{\text{VB}}) + \mathcal{H}(Q^{\text{CG}}) + \sum_{\mathbf{z}^{\text{VB}}, \mathbf{z}^{\text{CG}}} Q(\mathbf{z}^{\text{VB}}) Q(\mathbf{z}^{\text{CG}}) \log P(\mathbf{z}^{\text{VB}}, \mathbf{z}^{\text{CG}}, \mathbf{x} | \mathbf{d}) \tag{15}$$

Analogous to section 3 we apply Jensen's inequality in order to bring the average over Q^{VB} inside the convex function $\log P(\mathbf{z}^{\text{VB}}, \mathbf{z}^{\text{CG}}, \mathbf{x})$, resulting in an expression which we shall denote as $\log P(Q^{\text{VB}}, \mathbf{z}^{\text{CG}}, \mathbf{x} | \mathbf{d})$ for obvious reasons. The log-probability only depends on counts, so by bringing the average inside we find that $\log P(Q^{\text{VB}}, \mathbf{z}^{\text{CG}}, \mathbf{x}, \mathbf{d})$ depends on the quantities,

$$\mathbb{E}[N_{wkj}]_{Q^{\text{VB}}} = \hat{N}^{\text{VB}}_{wj} Q^{\text{VB}}_{k|wj} + \sum_{i \in \mathcal{S}^{\text{CG}}} z_{ik} x_{iw} d_{ij} \tag{16}$$

where $\hat{N}^{\text{VB}}_{wj}$ are the observed counts for word-type w and document j which are in the set $\mathcal{S}^{\text{VB}}$. In terms of this we

further need,

$$\mathbb{E}[N_{wk}]_{Q^{\text{VB}}} = \sum_j \hat{N}^{\text{VB}}_{wj} Q^{\text{VB}}_{k|wj} + \sum_{i \in \mathcal{S}^{\text{CG}}} z_{ik} x_{iw} \quad (17)$$

$$\mathbb{E}[N_{jk}]_{Q^{\text{VB}}} = \sum_w \hat{N}^{\text{VB}}_{wj} Q^{\text{VB}}_{k|wj} + \sum_{i \in \mathcal{S}^{\text{CG}}} z_{ik} d_{ij} \quad (18)$$

$$\mathbb{E}[N_k]_{Q^{\text{VB}}} = \sum_{wj} \hat{N}^{\text{VB}}_{wj} Q^{\text{VB}}_{k|wj} + \sum_{i \in \mathcal{S}^{\text{CG}}} z_{ik} \quad (19)$$

These expressions elegantly split the counts into a part described through a non-random mean field plus a sum over the remaining random variables that represent the fluctuations.

Thus, after applying Jensen's inequality we end up with the following lower bound on the evidence,

$$\mathcal{B} = \mathcal{H}(Q^{\text{VB}}) + \mathcal{H}(Q^{\text{CG}}) + \sum_{\mathbf{z}^{\text{CG}}} Q(\mathbf{z}^{\text{CG}}) \log P(Q^{\text{VB}}, \mathbf{z}^{\text{CG}}, \mathbf{x}|\mathbf{d}) \leq \mathcal{E} \quad (20)$$

Now let's assume we have drawn a sample from Q^{CG}, which we will denote with $\mathbf{z}^{\text{CG}}_s$. Furthermore, denote with $\bar{N}^s_{wjk}$ the value of $\mathbb{E}[N_{wjk}]_{Q^{\text{VB}}}$ evaluated at $\mathbf{z}^{\text{CG}}_s$ (see Eqn.16). Given this sample we bound again through linearization (see Eqn.10) which results in the following update for Q^{VB},

$$Q^{\text{VB}}_{k|wj} \propto \frac{\exp(\psi(\bar{N}^s_{wk} + \beta))}{\exp(\psi(\bar{N}^s_k + W\beta))} \exp(\psi(\bar{N}^s_{jk} + \alpha_k)) \quad (21)$$

In case we decide to use more than 1 sample we replace the expressions $\psi(\cdot) \rightarrow \langle \psi(\cdot) \rangle$, where the brackets $\langle \cdot \rangle$ denote taking the sample average.

The update for Q^{CG} will be sample-based. We first compute the variational update for Q^{CG},

$$Q^{\text{CG}} \propto P(Q^{\text{VB}}, \mathbf{z}^{\text{CG}}, \mathbf{x}|\mathbf{d}) \quad (22)$$

and subsequently draw samples from it. On closer inspection we see that this distribution is identical to the collapsed distribution $p(\mathbf{x}, \mathbf{z}|\mathbf{d})$, but over fewer data-cases, namely those in the set $\mathcal{S}^{\text{CG}}$, and with new effective hyper-parameters given by,

$$\alpha'_{jk} = \alpha_k + \sum_w \hat{N}^{\text{VB}}_{wj} Q^{\text{VB}}_{k|wj} \quad (23)$$

$$\beta'_{wk} = \beta + \sum_j \hat{N}^{\text{VB}}_{wj} Q^{\text{VB}}_{k|wj} \quad (24)$$

Hence, we can apply standard collapsed Gibbs sampling to draw from Q^{CG},

$$p(\mathbf{z}^{\text{CG}}_i = k|\mathbf{z}^{\text{CG}}_{\neg i}, \mathbf{x}, \mathbf{d}) \propto \frac{(\bar{N}^{\neg i,s}_{wk} + \beta)}{(\bar{N}^{\neg i,s}_k + W\beta)} (\bar{N}^{\neg i,s}_{jk} + \alpha_k) \quad (25)$$

These updates converge in expectation and stochastically maximize the expression for the bound on the evidence. In theory one should draw infinitely many samples from Q^{CG} to guarantee convergence. In practice however we have obtained very good results with drawing only a single sample before proceeding to the VB update.

It is also possible to infer the hyper-parameters $\{\alpha_k, \beta\}$ by either using sampling (Teh et al., 2004) or maximization (Minka, 2000). We refer to those papers for further details.

5.1 Extension To Collapsed Variational LDA

In (Teh et al., 2006) an improved variational approximation was proposed that operates in the same collapsed space as collapsed Gibbs sampling. This algorithm uses a factorized distribution $Q^{\text{CVB}} = \prod_i Q(z_i)$ but does not move this inside the log-probability as we did for SVB. Instead, it evaluates the necessary averages in the updates by assuming that the counts behave approximately normal and using a second order Taylor expansion around the mean. It was shown that including this second order information improves the standard VB approximation considerably.

This algorithm is also straightforwardly hybridized with collapsed Gibbs sampling by simply replacing the SVB updates with CVB updates in the hybrid SVB/CGS algorithm and using the same definitions for the counts as in Eqn.16. We call this algorithm CVB/CGS.

6 Extension to Bayesian Networks

The extension to collapsed Bayesian networks is relatively straightforward. First let's generalize SVB to Bayes nets. The derivation which includes variational distributions for the CPTs can be found in (Beal, 2003). We follow an alternative derivation that facilitates the transition to the hybrid SVB/CGS algorithm. We first collapse the state space by marginalizing out the CPTs. This results in an expression for the evidence that consists of products of factors, where each factor is a ratio of gamma-functions and the factors follow the structure of the original Bayes net. For instance, consider a hidden variable z which can assume state values k with two parents u, v which take values l, m respectively, see Fig.2. The factor associated with this family that will appear in the joint collapsed probability distribution of the BN is given by

$$F(\{z_i, u_i, v_i\}) = \prod_{lm} \left[\frac{\Gamma(\sum_k \alpha_k)}{\Gamma(N_{lm} + \sum_k \alpha_k)} \prod_k \left[\frac{\Gamma(N_{klm} + \alpha_k)}{\Gamma(\alpha_k)} \right] \right] \quad (26)$$

The complete joint (collapsed) probability distribution is given by a product of such factors, one for each family in the BN. This implies that the collapsed distribution inherits the graphical structure of the original BN, in particular its

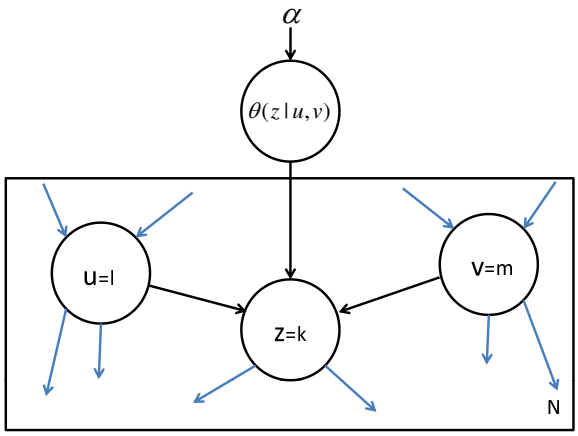


Figure 2: Family inside a BN consisting of two parents and one child.

treewidth, implying that collapsed inference using SVB is equally expensive as ordinary inference (for given CPTs) in the original BN.

Next, we move the average over the variational distribution $Q(\{\mathbf{z}_i\})$ inside the gamma-factors. This again produces a bound on the evidence. We then linearize the log-gamma factors around the current value of Q. Ignoring constant factors this results in the following terms for the family above,

$$G(Q, Q^*) = \text{terms for other families} + \sum_{klm} \left(\psi(\bar{N}_{klm} + \alpha_k) - \psi(\bar{N}_{lm} + \sum_k \alpha_k) \right) \bar{N}^*_{klm} \quad (27)$$

We recall again that $\bar{N}_{klm} = \sum_{w..} \hat{N}_{w..} Q_{klm|w..}$ where $w..$ represents all observed labels and $Q_{klm|w..}$ is the posterior marginal distribution over the variables z, u, v given observations.

All of these factors are local in the cliques of the original Bayes net. Hence, the update for Q becomes proportional to the product of local factors of the form,

$$F_{klm} = \frac{\exp(\psi(\bar{N}_{klm} + \alpha_k))}{\exp(\psi(\bar{N}_{lm} + \sum_k \alpha_k))} \quad (28)$$

As a result we can recompute new values for the local posterior marginals by running belief propagation over the junction tree associated with the original BN. Since $Q_{klm|w..}$ depends on $w..$, this has to be done for every combination of observed labels that has been observed at least once in the data (i.e. $\hat{N}_{w..} > 0$). It's interesting that the algorithm can thus be interpreted as an iterative belief propagation algorithm on a temporary graphical model where the potentials change from one iteration to the next. It bears a strong resemblance with iterative proportional fitting in which scaling updates enforce the constraints and alternate with message passing. In this interpretation, one could view the relations $\sum_{klm} N_{klm|w..} = \hat{N}_{w..}$ (which are equivalent to normalization of the $Q_{klm|w..}$) to be the constraints.

An alternative to VB is collapsed Gibbs sampling. Here one updates all hidden variables for a single data-item. One first removes the data-case from the pool and computes the expression for $p(\mathbf{z}_i|\mathbf{z}^{\neg i}, \mathbf{x}, \mathbf{d})$ over all hidden variables $\mathbf{z}$ in the Bayes net. This expression also factorizes according to the structure of the BN but the factors are now given by,

$$F'_{klm} = \frac{N^{\neg i}_{klm} + \alpha_k}{N^{\neg i}_{lm} + \sum_k \alpha_k} \quad (29)$$

One can draw samples by starting out at the leafs of the junction tree and computing distributions for the current node conditioned on upstream variables but marginalizing over all downstream variables. Given these variables we can then run an ancestral sampling pass outwards, back to the leaf nodes. This algorithm is an extension of the forward-filtering-backwards-sampling (FFBS) algorithm proposed in (Scott, 2002) for HMMs.

The derivation for the hybrid algorithm goes along similar lines as for LDA. We split all data-cases into two subsets, $\mathcal{S}^{\text{VB}}$ and $\mathcal{S}^{\text{CG}}$. We use Jensen's inequality to move the variational distribution Q^{VB} inside the log-gamma functions. Finally, we derive updates for Q^{VB} through the linearization trick. The final algorithm thus rotates over all data-cases, running either BP on the associated junction tree if the data-case is in $\mathcal{S}^{\text{VB}}$ (updating the Q^{VB} for all families of the BNs) or the FFBS algorithm if the data-case is in $\mathcal{S}^{\text{CG}}$. The expressions for the counts are always the analogs of those in Eqn.16.

7 Experiments

We report results on two datasets: 1) "KOS", which is harvested from a lefty blog site "*www.dailykos.com*"[1] and 2) "NIPS" which is a corpus of 2,484 scientific papers from the proceedings of NIPS[2]. KOS has $J = 3430$ documents, a vocabulary size of $W = 6909$, a total of $N = 467714$ words, and $M = 360664$ unique word-document pairs. NIPS has $J = 1740$, $W = 12419$, $N = 2166029$ and $M = 836644$. We used $K = 10$ for KOS and $K = 40$ for NIPS and set $\alpha_k = \beta = .1$ for both datasets.

In all sets of experiments, we withheld a random 10% of the words in the corpus for testing, while training on the remaining 90%. We compared a variety of algorithms: collapsed Gibbs sampling (CGS), standard VB (SVB), collapsed VB (CVB), as well as two hybrid algorithms: a hybrid of standard VB and CGS (SVB/CGS) as described in

[1]Downloadable from *http://yarra.ics.uci.edu/kos/*. Thanks to Dave Newman for pointing us to this site.

[2]Originally from *http://books.nips.cc* and preprocessed by Sam Roweis and Dave Newman.

Section 5, and a hybrid of collapsed VB (CVB/CGS) as described in Section 5.1. For both hybrid algorithms we only sampled data-cases for which $\hat{N}_{wj} = 1$. For all algorithms we trained for 300 iterations, testing after every iteration. In the figure captions we report the number of runs over which we averaged the results.

The algorithms were tested using the standard measure of individual word perplexity on the withheld test set. For the pure variational algorithms this is:

$$p(\{x_i^{\text{test}}\}|\{d_i^{\text{test}}\}) = \prod_i \sum_k \frac{\alpha_k + \bar{N}_{kd_i^{\text{test}}}}{\sum_k \alpha_k + \bar{N}_{d_i^{\text{test}}}} \frac{\beta + \bar{N}_{x_i^{\text{test}} k}}{W\beta + \bar{N}_k} \quad (30)$$

For CGS and the hybrid algorithms, we perform an online average over the samples after every iteration of sampling, discarding an initial burn in phase of 10 iterations,

$$p(\{x_i^{\text{test}}\}|\{d_i^{\text{test}}\}) = \prod_i \sum_k \frac{1}{S} \sum_{s=1}^{S} \frac{\alpha_k + \bar{N}^s_{kd_i^{\text{test}}}}{\sum_k \alpha_k + \bar{N}^s_{d_i^{\text{test}}}} \frac{\beta + \bar{N}^s_{x_i^{\text{test}} k}}{W\beta + \bar{N}^s_k} \quad (31)$$

The results for KOS and NIPS are shown in Figures 3, 4, 5 and 6. The variational algorithms converged faster than CGS or the hybrid algorithms, but converged to sub-optimal points. Collapsed algorithms performed better than the standard counterparts. The hybrid algorithms significantly improved upon the corresponding pure variational algorithms, with the performance of CVB/CGS being basically on par with CGS. To study how these results depend on the vocabulary size, we first ordered all the word-types according to their total number of occurrences and then only retained the top 3000 most frequent words for KOS and the top 6000 most frequent words for NIPS. The results are shown in Figures 7 and 8. Similar results were obtained with reduced vocabulary sizes of 4000 for KOS and 4000 and 8000 for NIPS. We conclude that for both datasets the perplexities have dropped significantly implying that prediction has become easier. However, the relative performance of the hybrid algorithms has not significantly changed.

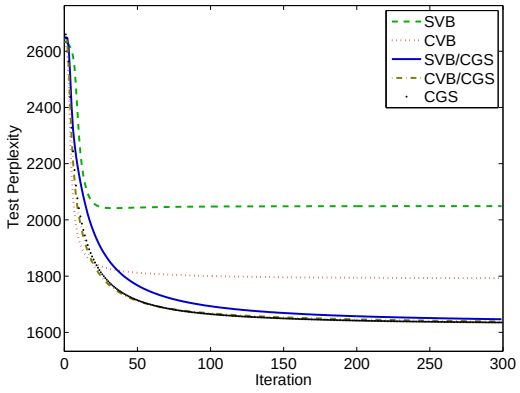


Figure 3: Perplexities of algorithms as function of number of iterations for KOS. Results averaged over 20 runs. The lines for CVB/CGS and CGS are on top of each other.

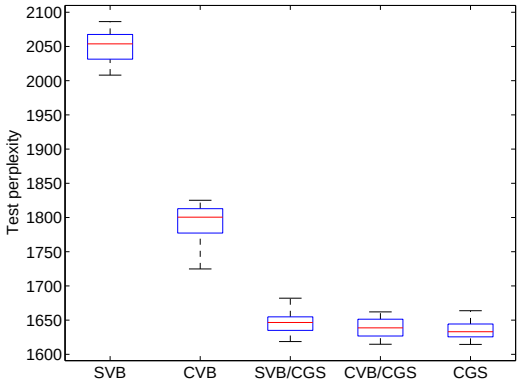


Figure 4: Final perplexities of algorithms at iteration 300 for KOS. Results averaged over 20 runs.

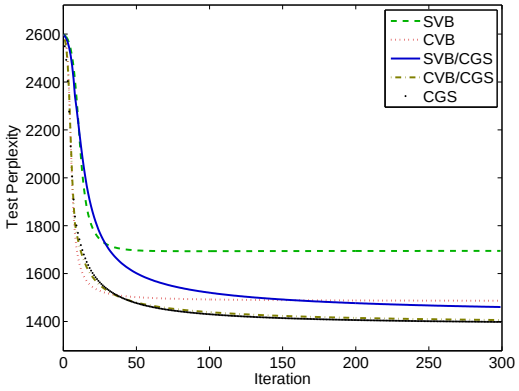


Figure 5: Perplexities of algorithms as function of number of iterations for NIPS. Results averaged over 17 runs.

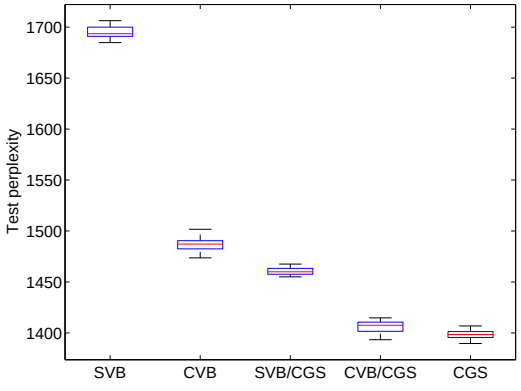


Figure 6: Final perplexities of algorithms at iteration 300 for NIPS. Results averaged over 17 runs.

To understand how much the algorithms learned from the singleton counts, $\hat{N}_{wj} = 1$, we first removed them from the training set but not from test set and subsequently re-

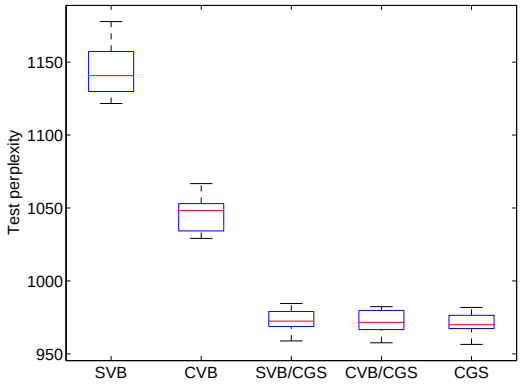


Figure 7: Final perplexities of algorithms at iteration 300 for KOS with a reduced vocabulary size of 3000 word types. Results averaged over 14 runs.

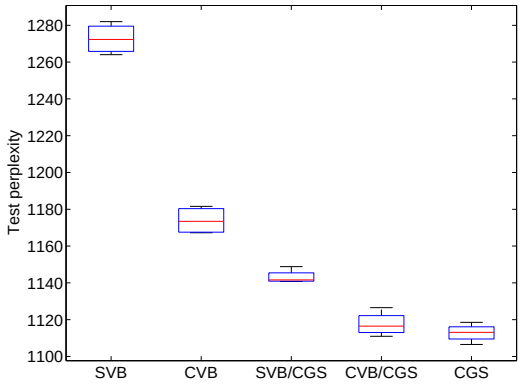


Figure 8: Final perplexities of algorithms at iteration 300 for NIPS with a reduced vocabulary size of 6000. Results averaged over 4 runs.

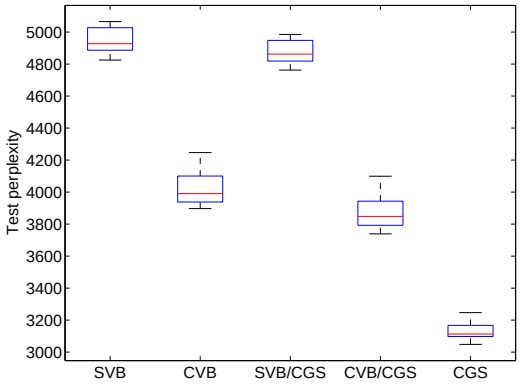


Figure 9: Final perplexities of algorithms at iteration 300 for KOS with all singleton counts removed from only training set. Results averaged over 20 runs.

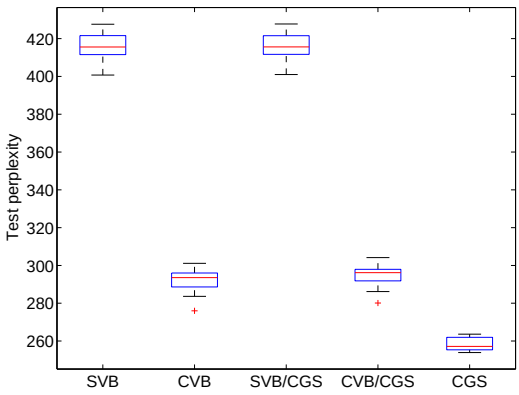


Figure 10: Final perplexities of algorithms at iteration 300 for KOS with all singleton counts removed from both training *and test* sets. Results averaged over 12 runs.

moved them from both training and test sets. The results for KOS are shown in Figures 9 and 10 (for a full vocabulary). In this case the difference between hybrid and VB algorithms is only due to the fact that we are still performing an online average for hybrid algorithms, even though we are not sampling any data-cases. We observe that the impact on the absolute values of the perplexities is very large indeed. The results for NIPS were qualitatively similar, but were much more moderate due to the fact that KOS contains many more very small counts than NIPS. We thus conclude that singleton counts play a very important, and sometimes even dominant role for text data in terms of perplexity. Whether this conclusion holds true for actual applications of LDA, such as indexing a corpus or retrieving similar documents to a test document remains to be seen.

For text data, where small counts are so dominant VB algorithms are not significantly faster than CGS. This is mainly due to the fact that the VB algorithms must compute[3] $exp(\psi(\cdot))$ which is more expensive than the simple count ratios necessary for CGS. For datasets which have relatively more large counts we expect the VB and hybrid algorithms to be significantly faster than CGS because they require only a single update per nonzero word-document entry rather then $\hat{N}_{wj}$ CGS updates for that entry.

8 Discussion and Related Work

In the context of topic models and Bayesian networks, we present a novel hybrid inference algorithm that combines variational inference with Gibbs sampling in a collapsed state space where the parameters have been marginalized out. We split the data-cases into two sets, $\mathcal{S}^{\text{CG}}, \mathcal{S}^{\text{VB}}$, where data-cases in the set $\mathcal{S}^{\text{CG}}$ are handled with Gibbs sampling while data-cases in the set $\mathcal{S}^{\text{VB}}$ are handled with variational updates. These updates interact in a consistent manner in that they stochastically optimize a bound on the evidence. In this paper we have restricted attention to discrete models, but extensions to the exponential family seem feasible.

[3] In fact, one could probably speedup the computation by noticing that $exp(\psi(\cdot))$ is almost linear for arguments larger than 5, but we haven't pursued this further.

The algorithm has the same flavor as stochastic EM algorithms where the E-step is replaced with a sample from the posterior distribution. Similarly to SEM (Celeux & Diebolt, 1985), the sequence of updates for Q_t^{VB} and $\mathbf{z}_t$ for the proposed algorithm forms a Markov chain with a unique equilibrium distribution, so convergence is guaranteed. How different this approximate equilibrium distribution is from the true equilibrium distribution remains to be studied.

The algorithm is also reminiscent of cutset sampling (Bidyuk & Dechter, 2007) where a subset of the nodes of a Bayesian network are sampled while the remainder is handled using belief propagation. It suggest an interesting extension of the proposed algorithm where one specifies a division of both data-cases *and nodes* into subsets $\mathcal{S}^{\text{CG}}$ and $\mathcal{S}^{\text{VB}}$. The nodes in $\mathcal{S}^{\text{VB}}$ should form a forest of low-treewidth junction trees for the algorithm to remain tractable. Alternatively, if the treewidth is too large, one can use loopy belief propagation on the set $\mathcal{S}^{\text{VB}}$. Our current choice for $\mathcal{S}^{\text{CG}}$ and $\mathcal{S}^{\text{VB}}$ was quite naive and motivated by our interest to understand why variational algorithm perform poorly for LDA. However, it seems preferable to develop more principled and perhaps adaptive (online) methods to make this division.

Still other hybridizations between variational and MCMC inference exist (de Freitas et al., 2001; Carbonetto & de Freitas, 2006) where variational distributions are used to guide and improve sampling. However, these algorithms solve a different problem and do not operate in collapsed state spaces.

To conclude, we believe that hybrid algorithms between the two major classes of inference schemes, namely variational and sampling, are a fruitful road to trade off bias, variance, computational efficiency and statistical accuracy.

Acknowledgements

Max Welling was supported by NSF grants IIS-0535278 and IIS-0447903 and by an NWO (Dutch Science Foundation) "Bezoekersbeurs". The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant BSIK03024.

A Lemma-1

Lemma-1: The following function is convex as a function of x,

$$z(x) = \sum_k \log\Gamma(x_k) - \log\Gamma(\sum_k x_k) \qquad (32)$$

Proof Lemma-1: We can write z(x) as a log-partition function as follows,

$$Z(x) = \exp(z(x)) = \int \mathrm{d}p_1 p_2..p_K \prod_k p_k^{x_k-1} = \frac{\prod_k \Gamma(x_k)}{\Gamma(\sum_k x_k)} \qquad (33)$$

It follows that the Hessian is equal to the covariance of $\{\log p_k\}$ and hence positive definite.

References

Beal, M. (2003). *Variational algorithms for approximate bayesian inference* (Technical Report). Gatsby Computational Neuroscience Unit, University College London, London, UK. PhD. Thesis.

Bidyuk, B., & Dechter, R. (2007). Cutset sampling for bayesian networks. *Journal Artificial Intelligence Research*, *28*, 1–48.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Buntine, W. (Ed.). (2002). *Variational extensions to em and multinomial pca*, vol. 2430 of *Lecture Notes in Computer Science*. Helsinki, Finland: Springer.

Carbonetto, P., & de Freitas, N. (2006). Conditional mean field. *NIPS*.

Castella, G., & Robert, C. (1996). Rao-blackwellisation of sampling schemes. *Biometrika*, *83(1)*, 81–94.

Celeux, G., & Diebolt, J. (1985). The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Comp. Statis. Quaterly*, *2*, 73–82.

de Freitas, N., d. F. R. Hojen-Sorensen, P. A., & Russell, S. J. (2001). Variational mcmc. *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence* (pp. 120–127).

Griffiths, T., & Steyvers, M. (2002). A probabilistic approach to semantic representation. *Proceedings of the 24th Annual Conference of the Cognitive Science Society*.

Heckerman, D. (1999). A tutorial on learning with bayesian networks. 301–354.

Minka, T. (2000). *Estimating a dirichlet distribution* (Technical Report).

Scott, S. L. (2002). Bayesian methods for hidden Markov models, recursive computing in the 21st century (pp. 337–351.).

Teh, Y., Jordan, M., Beal, M., & Blei, D. (2004). Hierarchical Dirichlet processes. *NIPS*.

Teh, Y., Newman, D., & Welling, M. (2006). A collapsed variational bayesian inference algorithm for latent dirichlet allocation. *NIPS*.

Teh, Y. W., Kurihara, K., & Welling, M. (2008). Collapsed variational inference for HDP. *Advances in Neural Information Processing Systems*.

Inference for Multiplicative Models

Ydo Wexler **Christopher Meek**

Microsoft Research, Microsoft Corporation
Redmond, WA 98052, USA

Abstract

The paper introduces a generalization for known probabilistic models such as log-linear and graphical models, called here multiplicative models. These models, that express probabilities via product of parameters are shown to capture multiple forms of contextual independence between variables, including decision graphs and noisy-OR functions. An inference algorithm for multiplicative models is provided and its correctness is proved. The complexity analysis of the inference algorithm uses a more refined parameter than the tree-width of the underlying graph, and shows the computational cost does not exceed that of the variable elimination algorithm in graphical models. The paper ends with examples where using the new models and algorithm is computationally beneficial.

1 Introduction

Probabilistic models that represent associations and/or interactions among random variables have been heavily applied in the past century in various fields of science and engineering. The statistical methods originating with the work of Fisher (1925, 1956) [6, 7] culminated in the log-linear models which describe the association patterns among a set of categorical variables without specifying any variable as a response (dependent) variable [1].

A specific type of probabilistic models, probabilistic graphical models, can be visually described as an interaction graph, and embody independence assumptions in the domain of interest [15]. Their main attraction is that the independences encoded in the structure of the model allow to indirectly specify the join distribution as a product of functions $\psi_i(D_i)$, each depends only on a limited set of variables D_i. Algorithms that compute the posterior distribution conditioned on evidence, called *inference* algorithms, exploit this structure, avoiding a direct computation of the join probabilities [5, 19]. The complexity of such algorithms depends on the topology of the model, and is exponential in the tree-width of the underlying graph.

The common distinction within graphical models is between undirected graphical models [15], a subset of log-linear models, where there are no restrictions on the functions ψ, and Bayesian networks (BNs) [19] in which every function is a conditional distribution $\psi_i(D_i) = P(X_i|\Pi_i)$ where Π_i is the set of parent variables of X_i in the model. Another type of probabilistic models that can be represented visually, called factor graphs, extends undirected graphical models and incorporates many of the desired properties of graphical modes [14].

Aside of the independences that are imposed by the model's structure, often there exist additional independences stemming from the specific values of the functions. These independences are not systematically exploited by the traditional inference algorithms, resulting in an unnecessary computational cost. For such non-structural independences we use the name context-specific independence (CSI), which was suggested in previous studies [2, 20]. We note that the term CSI takes here a more general meaning as it is not restricted to any specific type of independence.

Several studies have suggested changes in the traditional representation of graphical models in order to capture context-specific independences. These include similarity networks suggested by Heckerman (1991) [12], multinets (Geiger & Heckerman 1996) [9], asymmetric influence diagrams (Fung and Shachter 1990) [8], and structured representations of the functions ψ based on decision trees (Boutilier et al. 1996 [2], Poole& Zhang 2003 [20]). Other studies resorted to revised representations for specific functions (e.g. Quickscore algorithm by Heckerman 1989 for noisy-OR functions [11]).

Although the new representations proved useful from an empirical view point, they lack the ability to encompass a wide variety of CSI. In addition, the theoretical complexity of inference using these representation remained a function only of the topology of the graph underlying the model.

In this paper we approach the problem of inference from a more general perspective. We introduce a set of models called multiplicative models in which the functions ψ that account for the dependency of variables are in a multiplicative representation, where a value of an instance is a product over a set of parameters. We show that multiplicative models generalize over log-linear models, factor graphs, and graphical models. In addition, we show that multiplicative models can capture multiple forms of CSI, including CSIs captured via decision trees, decision graphs, and via noisy-OR functions. This leads to the question whether an inference algorithm that takes advantage of these independences can be constructed without additional cost. We provide such an algorithm, and show how different types of independences are utilized in this procedure to reduce the needed computations. The inference algorithm provided herein simplifies over the inference algorithm suggested by Poole & Zhang (2003) [20] when applied to Bayesian networks, by avoiding the use of tables and tables splitting operations. The more general nature of the algorithm also enables it to deal with different representations, and thus account for CSI that can not be represented by decision trees and decision graphs.

We prove the correctness of the inference procedure and give a new notion of complexity instead of the exponent of the tree-width which is commonly used to describe the complexity of inference in graphical models. The new time complexity is shown to be less than or equal to the standard complexity.

2 Multiplicative models

We propose a generalization of graphical models, factor graphs and log-linear models which represents the dependency of variables in the model via the notion of multiplicative models. In these models a value of an instance in the dependency function is a product over a specific set of parameters. The definition relies on the concept of a lattice. A lattice $(L, \trianglelefteq, \cap, \cup)$ is a partially ordered set (poset) with respect to some relation $\trianglelefteq$, in which for every two elements $l_1, l_2 \in L$ their least upper bound is denoted as $l_1 \cap l_2$ and their greatest lower bound is denoted as $l_1 \cup l_2$.

We usually use upper case letters to denote random variables and sets of random variables, and lower case letters to denote their values. For a variable V we denote its domain, or the set of possible values it can get, by $dom(V)$. For a set of variables $D = \{V_i\}_{i=1}^n$, the notation $dom(D)$ corresponds to the cross product of the domains $dom(V_i)$, $i = 1, \ldots, n$.

Let $D = \{V_i\}_{i=1}^n$ be a set of n multivalued variables, and let the function $\psi(D) : dom(D) \to \mathbb{R}$ specify the values in a full table for the set D, then the following is a definition for a mapping function of D.

Definition 1 (Mapping function) *A function f is called a mapping function of D with respect to the lattice L, if it is defined as $f : dom(Z) \to L$ for every $Z \subseteq D$, and maps partial instances $Z = z$ onto L.*

We use this definition to define a lattice multiplicative model of $\psi(D)$.

Definition 2 (Lattice multiplicative model) *A model $\rho = \{S_\rho, \Gamma_\rho\}$ of a function $\psi(D)$ is called a lattice multiplicative model with respect to a lattice $(L, \trianglelefteq, \cap, \cup)$ and a mapping function f, if $S_\rho \subseteq L$, $\Gamma_\rho = \{\gamma_s \in \mathbb{R} : s \in S_\rho\}$ and $\psi(D = d) = \prod_{s \trianglelefteq f(d), s \in S_\rho} \gamma_s$.*

The set S_ρ is called the structure of the model, and the set Γ_ρ is called the parameters of the model. In multiplicative models elements $s \in S$ for which $\gamma_s = 1$ can be removed from S.

Here we focus on a lattice L which is a set of propositional clauses over the variables and their values, and call this model a propositional multiplicative model, or simply a multiplicative model. In this model, the operators on the lattice are $\wedge$ and $\vee$. The mapping function used for this model is called the propositional mapping function, and is defined as follows.

Definition 3 (Propositional mapping function) *A mapping function f is called a propositional mapping function of D with respect to the lattice L, if for every set $Z \subseteq D$ the function maps every partial instance $Z = z$ into the conjunction $\bigwedge_{V_i \in Z} (V_i = v_i)$, where v_i is the projection of z onto the variable V_i.*

Definition 4 (Propositional multiplicative model) *A lattice multiplicative model $\rho = \{S_\rho, \Gamma_\rho\}$ of a function $\psi(D)$ is called a propositional multiplicative model with respect to a lattice $(L, \preceq, \wedge, \vee)$ and a propositional mapping function f, if the elements of L are propositional clauses over the variables in D and for two clauses c and c' we denote $c \preceq c'$ if c is implied by c'.*

Example 1 *Consider a set D which contains two ternary variables A and B. The corresponding lattice*

contains propositional clauses over A and B, and for the two clauses $c = (A = 0)$ and $c' = (A = 0) \wedge (B = 2)$ we denote $c \preceq c'$. The corresponding mapping function maps the instance $A = 0, B = 2$ into the propositional clause $(A = 0) \wedge (B = 2)$, and the partial instance $A = 0$ into the clause $(A = 0)$.

In this definition, the standard model which uses full-table representations of the functions $\psi(D)$, such as graphical models, and handles each instance separately, is also a multiplicative model with the set S containing all mapping $f(d)$ of instances $D = d$, and with values $\gamma_d = \psi(d)$.

Another well-known model that falls into Definition 2 is the log-linear model.

2.1 Log-linear models

Log-linear models are usually used to analyze categorical data, and are a direct generalization of undirected graphical models. These models that have been heavily used for statistical analysis for the past four decades describe the association patterns among a set of categorical variables without specifying any variable as a response (dependent) variable, treating all variables symmetrically [1].

Formally, a log-linear model specifies the natural log of the expected frequency of values d for a set of variables D as a linear combination of the main effect $\lambda_{v_i}^{V_i}$ of every variable $V_i \in D$, and if $|D| > 1$ interaction effects λ_s^S of every subset of variables $S \subseteq D$, where the instances s are consistent with d. For example, suppose that we want to investigate relationships between three categorical variables, A, B and C, then the full log-linear model is

$$\ln(F_{a,b,c}) = \mu + \lambda_a^A + \lambda_b^B + \lambda_c^C + \lambda_{ab}^{AB} + \lambda_{ac}^{AC} + \lambda_{bc}^{BC} + \lambda_{abc}^{ABC}$$

where μ is the overall mean of the natural log of the expected frequencies.

Clearly in the log-linear models instances are partially ordered by inclusion of their sets and by consistency of instantiations. To formalize log-linear models as a multiplicative models, for every subset $Z \subseteq D$ and for every instantiation $Z = z$ such that $\lambda_z^Z \neq 0$, the set S contains all clauses of the form $\bigwedge_{V \in Z} (V = v)$, where v is the projection of z onto the variable V. In addition, we set the parameters of the model to $\gamma_\top = e^\mu$ and $\gamma_{f(s)} = e^{\lambda_s^S}$.

2.2 Context-specific independence

With the introduction of graphical models and in particular Bayesian Networks (BNs), and the proof that inference in these models is NP-hard [4], several studies looked for further independences encoded in models that can potentially reduce the amount of work needed for inference [12, 9]. The notion of Context-Specific Independence (CSI) was then introduced by Smith et al. (1993) [23] and Boutilier et al. (1996) [2]. Context-specific independence corresponds to regularities within probabilistic models based on the values assigned in the model.

Formally, we say that the sets of variables X and Y are contextually independent in the context of $C = c$ given Z if

$$P(X, Y|Z = z, C = c) = \quad (1)$$
$$P(X|Z = z, C = c) \cdot P(Y|Z = z, C = c)$$

for every value $Z = z$. One aspect of this equation is that if X and Y are contextually independent given Z, then

$$P(X|Y = y_1, Z = z, C = c) = P(X|Y = y_2, Z = z, C = c) \quad (2)$$

for any two values y_1, y_2 of Y, which appear as repetitive values in conditional probability tables, such as those used in BNs. These repetition which are the basis of compact representations like decision trees and graphs were exploited for inference in BN [2, 20].

Another kind of CSI which was exploited for enhanced inference in BNs is the independence in noisy-OR functions. A noisy-OR function is a conditional probability function of a binary effect variable E given a set of m binary cause variables $C = \{C_1, \ldots, C_m\}$. The conditional probabilities of the function are $P(E = 0|C_1, \ldots, C_m) = c_0 \prod_{i:C_i=1} P(E = 0|C_i)$, where c_0 is a constant, and the values $P(E = 0|C_i)$ are some real numbers.

For any particular CSI of the sets of variables X and Y in the context $C = c$ given the set Z, as in Eq. 1, there exists a multiplicative model that captures this independence. Such a model is any multiplicative model where the structure does not contain elements s that involve variables from X and Y, such that there exists an instance $Z = z$ for which $s \wedge (Z = z) = \bot$ and $s \wedge (C = c) \neq \bot$.

We now define two types of multiplicative models that capture two different types of common CSIs.

2.2.1 Positive models

Representing the dependency of variables using log-linear models has some desirable properties, such as being general while ensuring the existence of a maximum likelihood without enforcing dependencies to be strictly positive. However, in the representation discussed in Section 2.1 the log-linear models use more parameters than necessary [3, 13]. Take for example the log-linear model for two binary variables A and B.

Assuming all possible effects exist, the corresponding log-linear model uses eight parameters rather than the four parameters in a standard representation as a full table: $\lambda_0^A, \lambda_1^A, \lambda_0^B, \lambda_1^B, \lambda_{00}^{AB}, \lambda_{01}^{AB}, \lambda_{10}^{AB}, \lambda_{11}^{AB}$.

Another representation of the log-linear models that accounts for these redundancies uses only parameters which involve non-zero instantiations of variables [10]. In the above example the only parameters used in this representation are: $\lambda_1^A, \lambda_1^B, \lambda_{11}^{AB}$. We describe this representation of log-linear models as a multiplicative model, which we call here the positive model.

Definition 5 (Positive model) *A positive model ρ of a function $\psi(D)$ is a multiplicative model wrt to the lattice $(L, \preceq, \wedge, \vee)$ and a (propositional) mapping function f in which S_ρ contains only elements $s = f(z)$ where $Z \subseteq D$ and no variable in $Z = z$ is set to zero.*

Log-linear models, and thus positive models, are known to capture conditional and contextual independences [16].

Example 2 *An example is a function ψ over two binary variables A and B where $\psi(0,0) \cdot \psi(1,1) = \psi(0,1) \cdot \psi(1,0)$. This implies that A is independent of B and the function can be written as $\psi(A,B) = \psi(A) \cdot \psi(B)$. In the corresponding positive model the parameter $\gamma_{(A=1)\wedge(B=1)} = \frac{\psi(0,0)\cdot\psi(1,1)}{\psi(0,1)\cdot\psi(1,0)} = 1$. Thus, this independence is captured in the model.*

Example 3 *In a more complex function with three binary variables A, B and C, every pair of variables is independent whenever the third variable is set to zero. For this function the corresponding positive model assigns $\gamma_{(V=1)\wedge(U=1)} = 1$ for every pair of variables $V, U \in \{A, B, C\}$ and where $V \neq U$.*

2.2.2 Decision trees and graphs as multiplicative models

Common structures for representing functions with contextual independence are decision trees (DTs) and decision graphs (DGs) [22, 18]. These structures capture contextual independences that are the result of repetitive values, as specified in Eq. 2. Several studies have used decision trees to enhance inference in graphical models [2, 20]. We show how DTs and DGs fall into the category of multiplicative models.

For a function $\psi(D)$ over a set of variables D, a decision tree T that represents $\psi(D)$ is a tree with variables from D at internal nodes and values from $\psi(D)$ at the leaves. Every edge from a variable V to a child in T corresponds to a different set of values $H \subseteq dom(V)$, and can be represented as a set of clauses $\bigvee_{v \in H} (V = v)$. A value at the end of a path $p = v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_m$, where v_i is some value of V_i, equals to the value of $\psi(d = v_1 v_2 \cdots v_m v_{m+1} \cdots v_n)$, where $V_j = v_j$ for $m < j \leq n$ is any possible value of V_j. We note that in a decision tree every instance $D = d$ is mapped to a single path in the tree. An example of a decision tree that encodes a function over the variables A, B, C, D is shown in Figure 1.

One can choose to use decision graphs [18] instead of decision trees. These are more compact structures that can encode for more distributions. For a function $\psi(D)$ over a set of variables D, a decision graph G that represents $\psi(D)$ is a directed graph with sets of variables from D at internal nodes and values from $\psi(D)$ at the leaves. Similar to decision trees, every edge from a set of variables W to a child Z corresponds to a different set of values $H \subseteq dom(W)$, and can be represented as a set of clauses $\bigvee_{w \in H} (W = w)$. A value at the end of a path p equals to the value of $\psi(d)$, where d is an instance of D consistence with the sets of values encoded by p. Again, as in decision trees, we note that in a decision graph every instance $D = d$ is mapped to a single path in the graph.

Definition 6 (Decision-graph model) *A decision graph model ρ of a function $\psi(D)$ is a multiplicative model wrt to the lattice $(L, \preceq, \wedge, \vee)$ and a mapping function f where every two elements $s_1, s_2 \in S_\rho$ satisfy $s_1 \wedge s_2 = \bot$, and $\bigvee_{s \in S} s = \top$, where $\bot = false$ and $\top = true$.*

For a specific decision graph G that represents $\psi(D)$, the decision graph model of G is $\rho(G)$ in which the structure contains one clause for every path from the root to a leaf in G, which is a conjunction of the clauses on the edges. For every such path s, we set γ_s to the value at the end of the path. We note that in this model for every instance $D = d$ there is only one element $s \in S_\rho$ such that $s \preceq f(d)$.

3 Inference for multiplicative models

Consider a model that encodes for the probability distribution $P(x) = \prod_i \psi_i(d_i)$, with sets $D_i = \{X_{i_1}, \ldots X_{i_{m_i}}\}$, and multiplicative models $\rho_i = \{S_i, \Gamma_i\}$ over all the functions $\psi_i(D_i)$ wrt a lattice $(L, \preceq, \wedge, \vee)$. We first show how to perform inference, and compute a probability of a set of query variables Q using a multiplicative model. In particular we perform inference for a multiplicative model via the variable elimination scheme (Zhang & Poole 1996 [24], Dechter 1999 [5]) which was originally suggested for inference in BNs. Then, we prove the correctness of the algorithm and analyze its time complexity.

We define an operation $M(V, \{\rho_i\})$, which given a variable $V \in X$ and a set of models $\{\rho_i\}$, $i = 1, \ldots, m$ over X returns a model ρ' over the variables $X \setminus V$. This

A	B	C	D	$\psi(A,B,C,D)$
0	0	0	0	0.4
0	0	0	1	0.4
0	0	1	0	0.4
0	0	1	1	0.4
0	1	0	0	0.8
0	1	0	1	0.8
0	1	1	0	0.8
0	1	1	1	0.8
1	0	0	0	0.1
1	0	0	1	0.1
1	0	1	0	0.032
1	0	1	1	0.08
1	1	0	0	0.1
1	1	0	1	0.1
1	1	1	0	0.65
1	1	1	1	0.08

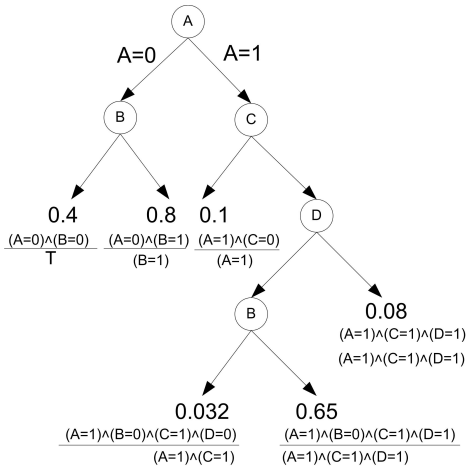


Figure 1: (left) A full-table over the binary variables A, B, C, D that specifies the value of the function ψ for each instance. (right) A decision tree corresponding to the function ψ on left. Under every node in the tree appears the corresponding proposition in the decision-tree representation, and below the corresponding proposition in a positive representation of the propositions in the decision tree.

operator is analogous to marginalization in standard inference algorithms. In addition, for a model ρ we define a relevance indicator $I_s(V)$ for each element $s \in S_\rho$ and each variable V in D, which is set to 1 if there exists a pair of instances d_1, d_2 of D that differ only by the value of V and for which $s \preceq f(d_1)$ but $s \not\preceq f(d_2)$. Otherwise, $I_s(V)$ is set to 0.

These operations allow us to write an inference procedure which computes the probability of a set of query variables in a multiplicative model as in Algorithm 1.

The algorithm operates like the bucket-elimination algorithm [5], where given an order on the variables we iterate over them (Line 2), and marginalize out one variable at a time (Line 9). Only elements that include terms that involve the current variable are considered in the marginalization.

Note that for graphical models, in which the elements of S_i are a mapping of instances of the functions D_i, this algorithm is exactly the known variable elimination algorithm, in its implementation as bucket-elimination [5], where the sets $S_i[j]$ are the tables in the bucket of the variable X_j.

A general algorithm for computing $M(V, \{\rho_i\})$ is given as Algorithm 2. We use there the notation $s \vee V$ for an element $s \in S$ and a variable V to denote $s \bigvee_{V=v} (V = v)$. This operation removes all terms that specify a value for V. For example, if $s = (V = 0) \wedge (U = 0)$ then $s \vee V = (U = 0)$.

The algorithm has two main parts: upto Line 5 the algorithm generates the set R of possible new elements in the model. From Line 6 it computes the new parameters γ_r, where at each iteration a "minimal" element of R is chosen, and selects those elements r with parameters $\gamma_r \neq 1$.

To compute the possible new elements, Lines 2 and 3 first create a closure under the operator $\wedge$ of each structure S_i. Then, in Line 5 all conjunctions of terms

Algorithm 1: VE for multiplicative models

Input: A model with n variables X_i $(i = 1, \ldots, n)$ and m functions $\psi_i(D_i \subseteq X)$, that encodes for the distribution $P(X)$. A set of multiplicative models $\rho_i = \{S_i, \Gamma_i\}$ wrt a mapping function f, where ρ_i model $\psi_i(D_i)$, and a set of k query variables $Q = \{X_i : i \leq k\}$

Output: The distribution $P(Q)$.

1 $t = m + 1$;
2 **for** $j = k + 1$ *to* n **do**
3 **for** $i = 1$ *to* $t - 1$ **do**
4 $S_i[j] \leftarrow \{s : s \in S_i \ , \ I_s(X_j) = 1\}$;
5 $\Gamma_i[j] \leftarrow \{\gamma_s : s \in S_i[j], \gamma_s \in \Gamma_i\}$;
6 $S_i \leftarrow S_i \setminus S_i[j]$;
7 $\Gamma_i \leftarrow \Gamma_i \setminus \Gamma_i[j]$;
8 end for;
9 $\{S_t, \Gamma_t\} \leftarrow M(V, \{S_i[j], \Gamma_i[j]\})$;
10 $t = t + 1$;
11 end for;
12 $P(Q) \leftarrow \left\{ P(q) = \prod_{s_i \preceq f(q)} \gamma_{s_i} : Q = q, s_i \in S_i \right\}$;
13 return $P(Q)$;

Figure 2: Algorithm for variable elimination with a multiplicative model

Algorithm 2: $M(V, \{\rho_i\})$

Input: A variable V and a set of representations $\rho_i = \{S_i, \Gamma_i\}$, $i = 1, \ldots, t$, wrt a lattice $(L, \preceq, \wedge, \vee)$, where $I_{s_i}(V) = 1$ for every i and $s_i \in S_i$.

Output: A representation $\rho' = \{S', \Gamma'\}$.

1 $S' \leftarrow \emptyset$; $\Gamma' \leftarrow \emptyset$;
2 **for** $i = 1$ *to* t **do**
3 $R_i = \{ \bigwedge_{s' \in S'_i} s' : S'_i \subseteq S_i\}$;
4 end for;
5 $R \leftarrow \{ \bigwedge_{1 \le i \le t} r_i : r_i \in R_i\}$;
6 **while** $R \neq \emptyset$ **do**
7 * *r is a minimal element in R* *
8 $r \in Min(R) = \{r' : r' \in R\ , \nexists r'' \in R \text{ s.t. } r'' \prec r'\}$;
9 $\gamma_r = \frac{\sum_{V=v} \prod_i \prod_{s \preceq (r \wedge (V=v)), s \in S_i} \gamma_s}{\prod_{r' \in S', r' \preceq r} \gamma_{r'}}$;
10 $R \leftarrow R \setminus \{r\}$;
11 **if** $\gamma_r \neq 1$ **then**
12 $S' \leftarrow S' \cup \{r\}$;
13 $\Gamma' \leftarrow \Gamma' \cup \{\gamma_r\}$;
14 end while;
15 return $\{S', \Gamma'\}$;

Figure 3: Algorithm for computing the operation $M(V, \{\rho_i\})$.

from the different closures consist of the set of possible new elements. In analogy to inference in graphical models, this operation is equivalent to the operation of tables' multiplication, often denoted as $\otimes$. In these models the set R is the set of instances in the table after marginalization.

We note that for some models, like graphical models, lines 2-5 are trivial, and are executed implicitly, since the elements in R are known to be all instances of a full-table over variables in $\bigcup S_i$.

3.1 Correctness of the inference procedure

We prove the correctness of Algorithm 1 by showing that the algorithm maintains the property that after iterating over the set of variables U, the models $\rho_i = \{S_i, \Gamma_i\}$ encode to the probability distribution $P(X \setminus U)$.

At the beginning of the algorithm every model ρ_i represents the corresponding function $\psi_i(D_i)$. Thus,

$$P(X = x) = \prod_i \psi_i(D_i = d_i) = \prod_i \prod_{s \preceq f(d_i), s \in S_i} \gamma_{i s}.$$

Assume that after removing the set of variable U we are left with the set $X' = X \setminus U$, and now wish to eliminate a variable $V \in X'$. We write the probability of an instance x'_v of $X' \setminus V$ which is the projection of an instance $X' = x'$ onto $X' \setminus V$ via the parameters γ:

$$P(x'_v) = \sum_{V=v} P(x') = \sum_{V=v} \prod_i \prod_{s \preceq f(x'), s \in S_i} \gamma_{i s}$$

We can decompose the product into terms that involve the variable V and those which do not. Denoting $\alpha(x'_v) = \prod_i \prod_{s \preceq f(x'), s \in S_i, I_s(V)=0} \gamma_{i s}$, we get

$$P(x'_v) = \alpha(x'_v) \cdot \sum_{V=v} \prod_i \prod_{s \preceq f(x'), s \in S_i, I_s(V)=1} \gamma_{i s}. \quad (3)$$

Now, lets examine what the algorithm encodes for after removing variable V, and show that it equals Eq. 3. While the elements that do not involve variable V are not changed, the elements that do involve V are removed and the elements $s_t \in S_t$ are added. Therefore, after applying Algorithm 2 for V the remaining sets encode for $\hat{P}(x'_v) = \alpha(x'_v) \cdot \beta(x'_v)$ where $\beta(x'_v) = \prod_{s_t \preceq f(x'), s_t \in S_t} \gamma_{s_t}$. To express $\beta(x'_v)$ in the terms of Algorithm 2, recall that $S_t \subseteq R$ and if an element $s \in R$ and $s \notin S_t$ then $\gamma_s = 1$. Thus, we can rewrite $\beta(x'_v)$ using elements of R as

$$\beta(x'_v) = \prod_{r \preceq f(x'), r \in R} \gamma_r.$$

From lines 2-5 in Algorithm 2, there is one element $r^* \preceq f(x')$ in R for which $\forall r \in R$ such that $r \preceq f(x')$ also satisfies $r \preceq r^*$. First, to show there is such an element r^* we recall from Line 5 that all elements in R can be written as $r = \bigwedge_{1 \le i \le t} r_i$, where $r_i \in R_i$, and R_i is the closure of S_i under the operator $\wedge$. Consider the set of elements $r_i^* \preceq f(x')$, $i = 1, \ldots, t$, for which all other elements $r_i \in R_i$ such that $r_i \preceq f(x')$ satisfy $r_i \preceq r_i^*$. Then, every element $r = \bigwedge_{1 \le i \le t} r_i$ such that $r \preceq f(x')$ also satisfies $r \preceq r^*$.

Now, assume by contradiction that there were two such elements, $r_1^*, r_2^* \in R$. Then from the definition of r_1^* and r_2^* we get $r_1^* \preceq r_2^*$ and $r_2^* \preceq r_1^*$, yielding $r_1^* = r_2^*$. Thus, from line 9 in Algorithm 2

$$\beta(x'_v) = \gamma_{r^*} \cdot \prod_{r \preceq r^*, r \in R} \gamma_r = \sum_{V=v} \prod_i \prod_{s \preceq (r^* \wedge (V=v)), s \in S_i} \gamma_s$$

where the last equality is due to the fact that the denominator in the computation of γ_{r^*} is $\prod_{r \preceq r^*, r \in R} \gamma_r$. In the terms of Algorithm 1 the set $\{s : s \preceq (r^* \wedge (V =$

$v)), s \in S_i\}$ can be rewritten as $\{s : s \preceq f(x'), s \in S_i, I_s(V) = 1\}$. Thus, we can write

$$\beta(x'_v) = \prod_i \prod_{s \preceq f(x'), s \in S_i, I_s(V)=1} \gamma_s$$

and from Eq. 3 we get $\hat{P}(x'_v) = P(x'_v)$. Namely, the new models encode for $P(X' \setminus V)$.

3.2 Incorporating evidence

In many practical scenarios we observe the value of some of the variables in the model, and wish to incorporate this evidence. The multiplicative models allow us to do so in a most natural way. Consider a set E of evidence nodes for which we observed the values $E = e$, and a multiplicative model $\rho = \{S_\rho, \Gamma_\rho\}$. Then, in order to incorporate the evidence into ρ, we adjust the elements in S_ρ by $s = s \bigwedge_{V \in E} (V = v)$, where v is the projection of e onto the variable $V \in E$. Then, we remove every element not consistent with the evidence, $s = \perp$.

3.3 Complexity of inference

It is well known that the complexity of inference in graphical models is NP-hard and its cost exponential in the tree-width of the underlying graph [4].

We analyze the time complexity of the inference procedure for multiplicative models given in Algorithm 1. As a by-product we refine the standard complexity and provide a new complexity bound which is based on the representation used. One can then say that the complexity of the problem is the minimum complexity among all possible representations.

3.3.1 Diameter of multiplicative models

The structure of a multiplicative model determines the amount of computations needed to obtain the value $\psi(d)$ of a single instantiation of values to variables in a set D. Although at first glance it seems that for a model $\rho = \{S, \Gamma\}$ of a function $\psi(D)$ the number of operations needed to obtain values of all instances $D = d$ amounts to a total of $\sum_{D=d} |\{s : s \preceq d\}|$, the real number of operations can be dramatically lower and we denote it by $\delta(\rho)$. For hierarchical models, in which if an element s is not in the structure of the model then all elements $s \preceq s'$ are also not in the model, Good (1963) provides a method that computes all such values in time $|S| \log |S|$ [10]. We denote the ratio between the number of computations and the number of elements in S, which is the size of the model, by $diam(\rho) = \frac{\delta(\rho)}{|S|}$ and name it the diameter of ρ.

From a computational perspective, it is clearly beneficial to use models with a small diameter, as this directly leads to fewer operations whenever we want to either obtain a value of ψ or update the values γ_s. Examples of models with a diameter of 1 are graphical models and decision graph models, in which for every element $s \in S$, the only element s' such that $s' \preceq s$, is s itself. On the other hand, the diameter of a positive model can be as high as $\frac{\log |S|}{2}$. This maximum is achieved for a positive model of m binary variables, when all 2^m parameters do not equal one, and hence all possible elements are in S. In this scenario the diameter is exactly $\frac{m}{2}$.

Although in the worst scenario the diameter of a positive model can be large, often this is not the case, and the diameter is typically bounded to be very small.

Example 4 *Consider as an example the Potts model [21] in which a function $\psi(D)$ over a set $D = \{V_i\}_{i=1}^n$ decomposes according to $\psi(D = d) = c_0 \prod_{i,j} \psi(v_i, v_j)$, where v_i and v_j are projections of d onto the variables V_i and V_j respectively, and c_0 is a constant. Although in general a positive model over n binary variables has a diameter of $\frac{n}{2}$, in this example, the structure of the positive model includes only elements that involve at most two variables. Therefore, the diameter of the model is bounded by two.*

Similarly, in a more complex scenario where the function ψ decomposes to functions of k-tuples of variables, the diameter will be bounded by k.

Consider a tree decomposition of the graph in which there is an edge between a pair of variables V, U if there exists an element s in one of the models for which $I_s(V) \cdot I_s(U) = 1$. We denote by $S(W) = \{s \vee (X \setminus Z) : s \in S_i\}$ the set of parts of elements in the models ρ_i that involve variables from the set of graph vertices Z which is mapped onto the tree node W. Further denoting as $S^-(W)$ the closure of $S(W)$ under the operator $\wedge$, we say that complexity of the algorithm for this tree decomposition is the maximum over the nodes W in the tree of $|S^-(W)| \cdot diam(S^-(W))$, as described in Section 3.3.1. Then, the overall complexity of the algorithm is the complexity for the tree decomposition that yields the minimum for this term.

To see that this is indeed the time complexity of the algorithm, consider the elements in a set R in Algorithm 2. The number of elements there does not exceed the number of elements in $S^-(W)$ for the corresponding tree decomposition and where W maps onto the variables that appear in R. Most of the computation stems from computing the products in Line 9, and these can be done for the entire set of elements of R in time proportional to $|R| \cdot diam(R)$. Therefore, having the ability to choose an elimination order, the complexity of the algorithm is $|S^-(W)| \cdot diam(S^-(W))$

maximized over all nodes W in a tree decomposition and minimized over all possible such decompositions.

3.4 Benefits of inference for multiplicative models

Different multiplicative models capture different contextual independences, hence specifying different number of parameters. Take for example the function over four binary variables A, B, C, D with values according to the table in Figure 1. The structure of the corresponding decision-tree model contains six elements while the structure of the corresponding positive model contains eight elements. In this latter model, the CSI captured in the decision tree, yielding the value of ψ to be independent of B given that A, C and D are set to one, does not have any effect. This variation and the structure of the model affect the run time of the inference algorithm.

An example where there are substantial computational savings when using the inference algorithm proposed can be found in a model such as the QMR-DT network [17], which is comprised of noisy-OR functions, mentioned in Section 2.2. The QMR-DT network is a two-level or bipartite BN where all variables are binary. The top level of the graph contains nodes for the diseases C, and the bottom level contains nodes for the findings E. The conditional probabilities in the network $P(E_i = e_i|\Pi_i)$, where Π_i are the parents of finding E_i in the network, are represented by noisy-OR functions.

Heckerman (1989) has developed an algorithm, called Quickscore, which takes advantage of the independence of the cause variables in the context of a negative finding $E_i = 0$ and uses it to speed up inference in the QMR-DT network [11].

For every noisy-OR function $P(E|C_1, \ldots, C_m)$ a structure of a multiplicative model that captures the independence does not contain elements s such that $s \wedge (E = 0) \neq \perp$ for which $I_s(C_i) = 1$ and $I_s(C_j) = 1$, for all $i, j \leq m$.

In addition, running Algorithm 1 using multiplicative models with structures

$$S_i = \{(E_i = 1) \bigwedge_{C_i \in \Pi_i} (C_i = c_i) : \forall C_i = c_i\} \vee$$

$$\{(E_i = 0) \wedge (C_i = 1) : C_i \in \Pi_i\} \vee ((E_i = 0) \bigwedge_{C_i \in \Pi_i} (C_i = 0))$$

is identical to the Quickscore algorithm and gains the same savings automatically.

References

[1] Y. Bishop, E. Fienberg, and P. Holland. *Discrete multivariate analysis*. MIT Press, 1975.

[2] C. Boutilier and et al. Context-specific independence in Bayesian networks. In *Uncertainty in Artificial Intelligence*, pages 115–123, 1996.

[3] R. Christensen. *Log-Linear Models and Logistic Regression*. Springer, 1997.

[4] G. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[5] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.

[6] R. Fisher. *Statistical Methods for Research Workers*. Macmillan Pub Co, 1925.

[7] R. Fisher. *Statistical Methods and Scientific Inference*. Oliver and Boyd, 1956.

[8] R. Fung and R. Shachter. Contingent influence diagrams. *Working Paper, Dept. of Engineering-Economic Systems, Stanford University*, 1990.

[9] D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and bayesian multinets. *Artificial Intelligence*, 82:45–74, 1996.

[10] I. Good. Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *The Annals of Math. Stat.*, 34:911–934, 1963.

[11] D. Heckerman. A tractable inference algorithm for diagnosing multiple diseases. *UAI*, 230:362–367, 1989.

[12] D. Heckerman. *Probabilistic Similarity Networks*. MIT Press, 1991.

[13] D. Knoke and P. Burke. *Log-Linear Models*. Sage Publications Inc, 1980.

[14] F. R. Kschischang, B. Frey, and H. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory*, 47(2):498–519, 2001.

[15] S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[16] J. Lindsey. Conditional independence and log linear models for multi-dimensional contingency tables. *Quality and Quantity*, 8(4):377–390, 1974.

[17] B. Middleton and et al. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base: Part II. Evaluation of diagnostic performance. *SIAM Journal on Computing*, 30:256–267, 1991.

[18] J. J. Oliver. Decision graphs - an extension of decision trees. *Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics*, pages 343–350, 1993.

[19] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[20] D. Poole and N. Zhang. Exploiting contextual independence in probabilistic inference. *JAIR*, 18:263–313, 2003.

[21] R. Potts. Some generalized order-disorder transformations. *Proceedings of the Cambridge Philosophical Society*, 48:106–109, 1952.

[22] S. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[23] J. Smith, S. Holtzman, and J. Matheson. Structuring conditional relationships in influence diagrams. *Oper. Res.*, 41(2):280–297, 1993.

[24] N. Zhang and D. Poole. Exploiting causal independence in bayesian network inference. *JAIR*, 5:301–328, 1996.

Refractor Importance Sampling

Haohai Yu and **Robert A. van Engelen**
Department of Computer Science
Florida State University
Tallahassee, FL 32306-4530 USA
{hyu,engelen}@cs.fsu.edu

Abstract

In this paper we introduce Refractor Importance Sampling (RIS), an improvement to reduce error variance in Bayesian network importance sampling propagation under evidential reasoning. We prove the existence of a collection of importance functions that are close to the optimal importance function under evidential reasoning. Based on this theoretic result we derive the RIS algorithm. RIS approaches the optimal importance function by applying localized arc changes to minimize the divergence between the evidence-adjusted importance function and the optimal importance function. The validity and performance of RIS is empirically tested with a large set of synthetic Bayesian networks and two real-world networks.

1 Introduction

The Bayesian Network (BN) [Pearl, 1988] formalism is one of the dominant representations for modeling uncertainty in intelligent systems [Neapolitan, 1990, Russell and Norvig, 1995]. A BN is a probabilistic graphical model of a joint probability distribution over a set of statistical variables. Bayesian inference on a BN answers probabilistic queries about the variables and their influence relationships. The posterior probability distribution is computed using belief updating methods [Pearl, 1988, Guo and Hsu, 2002]. Exact inference is NP-hard [Cooper, 1990]. Thus, exact methods only admit relatively small networks or simple network configurations in the worst case. Approximations are also NP-hard [Dagum and Luby, 1993]. However, approximate inference methods have anytime [Garvey and Lesser, 1994] and/or anywhere [Santos et al., 1995] properties that make these methods more attractive compared to exact methods.

Stochastic simulation algorithms, also called stochastic sampling or Monte Carlo (MC) algorithms, form one of the most prominent subclasses of approximate inference algorithms of which Logic Sampling [Henrion, 1988] was the first and simplest sampling algorithm. Likelihood weighting [Fung and Chang, 1989] was designed to overcome the poor performance of logic sampling under evidential reasoning with unlikely evidence. Markov Chain Monte Carlo (MCMC) forms another important group of stochastic sampling algorithms. Examples in this group are Gibbs sampling, Metropolis sampling and hybrid-MC sampling [Geman and Geman, 1984, Gilks et al., 1996, MacKay, 1998, Pearl, 1987, Chavez and Cooper, 1990]. Stratified sampling [Bouckaert, 1994], hypercube sampling [Cheng and Druzdzel, 2000c], and quasi-MC methods [Cheng and Druzdzel, 2000b] generate random samples from uniform distributions using various methods to improve sampling results. The importance sampling methods [Rubinstein, 1981] are widely used in Bayesian inference. Self Importance Sampling (SIS) [Shachter and Peot, 1990] and Adaptive Importance Sampling (AIS-BN) [Cheng and Druzdzel, 2000a] are among the most effective algorithms.

In this paper we prove that the importance functions of an evidence-updated BN can only approach the optimal importance function when the BN graph structure is modified according to the observed evidence. This implies the existence of a collection of importance functions with minimum divergence to the optimal importance function under evidential reasoning. Based on this result we derive our Refractor Importance Sampling (RIS) class of algorithms. In contrast to AIS-BN and SIS methods, RIS removes the lower bound that prevents the updated importance function to approach the optimal importance function. This is achieved by a graphical structure "refractor", consisting of a localized network structure change that minimizes the divergence between the evidence-adjusted importance function and the optimal importance function.

The remainder of this paper is organized as follows. Section 2 proves the existence of a lower bound on the divergence to the optimal importance function under evidential reasoning with a BN. The lower bound is used to derive the class of RIS algorithms introduced in Section 3. Section 4 empirically verifies the properties of the RIS algorithms on a large set of synthetic networks and two real-world networks, and compares the results to other importance sampling algorithms. Finally, Section 5 summarizes our conclusions and describes our future work.

2 Importance Function Divergence

In this section we first give BN definitions and briefly review importance sampling. We then give a KL-divergence lower bound for importance sampling error variance. We prove the existence of a collection of importance functions that approach the optimal importance function by adjusting both the quantitative and qualitative components of a BN under dynamic updating with evidence.

2.1 Definitions

The following definitions and notations are used.

Def. 1 *A* Bayesian network $BN = (G, \Pr)$ *is a DAG* $G = (\mathbf{V}, \mathbf{A})$ *with vertices* $\mathbf{V}$ *and arcs* $\mathbf{A}$, $\mathbf{A} \subseteq \mathbf{V} \times \mathbf{V}$. $\Pr$ *is the joint probability distribution over the discrete random variables (vertices)* $\mathbf{V}$ *defined by* $\Pr(\mathbf{V}) = \prod_{V \in \mathbf{V}} \Pr(V \mid \pi(V))$. *The set of parents of a vertex* V *is* $\pi(V)$. *The* conditional probability tables *(CPT) of the BN assign values to* $\Pr(V \mid \pi(V))$ *for all* $V \in \mathbf{V}$.

The graph G induces the *d-separation criterion* [Pearl, 1988], denoted by $\langle \mathbf{X}, \mathbf{Y} \mid \mathbf{Z} \rangle$, which implies that $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent in $\Pr$ given $\mathbf{Z}$, with $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$.

Def. 2 *Let* $BN = (G, \Pr)$ *be a Bayesian network.*

- *The* combined parent set *of* $\mathbf{X} \subseteq \mathbf{V}$ *is defined by* $\boldsymbol{\pi}(\mathbf{X}) = \bigcup_{X \in \mathbf{X}} \pi(X) \setminus \mathbf{X}$.
- *Let* $An(\cdot)$ *denote the transitive closure of* $\pi(\cdot)$, *i.e. the ancestor set of a vertex. The* combined ancestor set *of* $\mathbf{X} \subseteq \mathbf{V}$ *is defined by* $\mathbf{An}(\mathbf{X}) = \bigcup_{X \in \mathbf{X}} An(X) \setminus \mathbf{X}$.
- *Let* $\delta : \mathbf{V} \rightarrow \mathbb{N}$ *denote a topological order of the vertices such that* $Y \in An(X) \rightarrow \delta(Y) < \delta(X)$. *The* ahead set *of a vertex* $X \in \mathbf{V}$ *given* δ *is defined by* $Ah(X) = \{Y \in \mathbf{V} \mid \delta(Y) < \delta(X)\}$.

2.2 Importance Sampling

Importance sampling is an MC method to improve the convergence speed and reduce the error variance with probability density functions. Let $g(\mathbf{X})$ be a function of m variables $\mathbf{X} = \{X_1, \ldots, X_m\}$ over domain $\Omega \subseteq \mathbb{R}^m$, such that computing $g(\mathbf{X})$ for any $\mathbf{X}$ is feasible. Consider the problem of approximating $I = \int_\Omega g(\mathbf{X}) d\mathbf{X}$ using a sampling technique. Importance sampling approaches this problem by rewriting $I = \int_\Omega \frac{g(\mathbf{X})}{f(\mathbf{X})} f(\mathbf{X}) d\mathbf{X}$, where $f(\mathbf{X})$ is a probability density function over Ω, often referred to as the importance function. In order to achieve minimum error variance equal to $\sigma^2_{f(\mathbf{X})} = (\int_\Omega |g(\mathbf{X})| d\mathbf{X})^2 - I^2$, the importance function should be $f(\mathbf{X}) = |g(\mathbf{X})| (\int_\Omega |g(\mathbf{X})| d\mathbf{X})^{-1}$, see [Rubinstein, 1981]. Note that when $g(\mathbf{X}) > 0$ the optimal probability density function is $f(\mathbf{X}) = g(\mathbf{X}) I^{-1}$ and $\sigma^2_{f(\mathbf{X})} = 0$. It is obvious that in most of cases it is impossible to obtain the optimal importance function.

The SIS [Shachter and Peot, 1990] and AIS-BN [Cheng and Druzdzel, 2000a] sampling algorithms are effective methods for approximate Bayesian inference. These methods attempt to approach the optimal importance function through learning by dynamically adjusting the importance function during sampling with evidence. To this end, AIS-BN heuristically changes the CPT values of a BN, a technique that has been shown to significantly improve the convergence rate of the approximation to the exact solution.

We use the following definitions for sake of exposition.

Def. 3 *Let* $BN = (G, \Pr)$ *be a Bayesian network with* $G = (\mathbf{V}, \mathbf{A})$ *and evidence* $\mathbf{e}$ *for variables* $\mathbf{E} \subseteq \mathbf{V}$. *A* posterior $BN_\mathbf{e}$ *of the BN is some (new) network defined as* $BN_\mathbf{e} = (G_\mathbf{e}, \Pr_\mathbf{e})$ *with graph* $G_\mathbf{e}$ *over variables* $\mathbf{V} \setminus \mathbf{E}$, *such that* $BN_\mathbf{e}$ *exactly models the posterior joint probability distribution* $\Pr_\mathbf{e} = \Pr(\cdot \mid \mathbf{e})$.

A typical example of a *posterior* $BN_\mathbf{e}$ is a BN combined with an updated posterior state as defined by exact inference algorithms, e.g. using evidence absorption [van der Gaag, 1996]. Approximations of $BN_\mathbf{e}$ are used by importance sampling algorithms. These approximations consist of the original BN with all evidence vertices ignored from further consideration.

Def. 4 *Let* $BN = (G, \Pr)$ *be a Bayesian network with* $G = (\mathbf{V}, \mathbf{A})$ *and evidence* $\mathbf{e}$ *for variables* $\mathbf{E} \subseteq \mathbf{V}$. *The* evidence-simplified $ESBN_\mathbf{e}$ *of BN is defined by* $ESBN_\mathbf{e} = (G'_\mathbf{e}, \Pr'_\mathbf{e})$, *where* $G'_\mathbf{e} = (\mathbf{V}'_\mathbf{e}, \mathbf{A}'_\mathbf{e})$, $\mathbf{V}'_\mathbf{e} = \mathbf{V} \setminus \mathbf{E}$, *and* $\mathbf{A}'_\mathbf{e} = \{(X, Y) \mid (X, Y) \in \mathbf{A} \bigwedge X, Y \notin \mathbf{E}\}$.

The joint probability distribution $\Pr'_\mathbf{e}$ of an evidence-simplified BN approximates $\Pr_\mathbf{e}$. For example, SIS and AIS-BN adjust the CPTs of the original BN.

2.3 KL-Divergence Bounds

We give a lower bound on the KL-divergence [Kullback, 1959] of the evidence-simplified $\Pr'_{\mathbf{e}}$ from the exact $\Pr_{\mathbf{e}}$. The lower bound is valid for all variations of $\Pr'_{\mathbf{e}}$, including those generated by importance sampling algorithms that adjust the CPT.

Theorem 1 *Let $ESBN_{\mathbf{e}} = (G'_{\mathbf{e}}, \Pr'_{\mathbf{e}})$ be an evidence-simplified BN given evidence $\mathbf{e}$ for $\mathbf{E} \subseteq \mathbf{V}$. If $\Pr'_{\mathbf{e}}(V \mid \pi_{\mathbf{e}}(V)) = \Pr(V \mid \pi(V), \mathbf{e}))$ for all $V \in \mathbf{V}$ then the KL-divergence between $\Pr_{\mathbf{e}}$ and $\Pr'_{\mathbf{e}}$ is minimal and given by*

$$\sum_{X \in \mathbf{X}} \sum_{Cfg(X, \pi(X))} \Pr(x, \pi(x) \mid \mathbf{e}) \ln \Pr(x \mid \pi(x)) + \sum_{X \in \mathbf{X}} \sum_{Cfg(X, \pi(X))} \Pr(x, \pi(x) \mid \mathbf{e}) \ln \frac{1}{\Pr'_{\mathbf{e}}(x \mid \pi_{\mathbf{e}}(x))} + \sum_{Cfg(\boldsymbol{\pi}(\mathbf{E}))} \Pr(\boldsymbol{\pi}(\mathbf{e}) \mid \mathbf{e}) \ln \prod_{e \in \mathbf{e}} \Pr(e \mid \pi(e)) - \ln \Pr(\mathbf{e}) \quad (1)$$

where $\mathbf{X} = \mathbf{V} \setminus \mathbf{E}$.

Proof. See Appendix A. □

Theorem 1 bounds the error variance from below, which is empirically verified for SIS and AIS-BN in the results Section 4. The divergence Eq. (1) is zero when specific conditions are met as stated below.

Corollary 1 *Let $ESBN_{\mathbf{e}} = (G'_{\mathbf{e}}, \Pr'_{\mathbf{e}})$ be an evidence-simplified BN given evidence $\mathbf{e}$ for $\mathbf{E} \subseteq \mathbf{V}$. If $\boldsymbol{\pi}(\mathbf{E}) \cap (\mathbf{V} \setminus \mathbf{E}) = \emptyset$, then $\Pr'_{\mathbf{e}} = \Pr_{\mathbf{e}}$.*

Proof. See Appendix B. □

Hence, the optimal importance function is obtained when all evidence vertices are clustered as roots in G.

We will now show how $\Pr'_{\mathbf{e}}$ can approach the optimal $\Pr_{\mathbf{e}}$ without restrictions. For sake of explanation, the following widely-held assumptions are reiterated:

Assumption 1 *The topological order δ of a BN and its* posterior *version $\delta_{\mathbf{e}}$ of $BN_{\mathbf{e}}$ are consistent. That is, $\delta_{\mathbf{e}}(Y) < \delta_{\mathbf{e}}(X) \rightarrow \delta(Y) < \delta(X)$ for all $X, Y \in \mathbf{V} \setminus \mathbf{E}$.*

Assumption 1 is reasonable for the following facts:

1. According to chain rule, a BN can be built up in any topological order and all of them describe the same joint probability distribution.
2. Although there has never been a widely accepted definition of what causality is, it is widely accepted that the fact of observing evidence for random variables should not change the causality relationship between the variables.

Theorem 2 *Let $BN_{\mathbf{e}}(G_{\mathbf{e}}, \Pr_{\mathbf{e}})$ be the* posterior *of a $BN = (G, \Pr)$ given evidence $\mathbf{e}$ for $\mathbf{E} \subseteq \mathbf{V}$. If $X \notin \mathbf{An}(\mathbf{E})$ for all $X \in \mathbf{V} \setminus \mathbf{E}$, then $\Pr_{\mathbf{e}}(X \mid Ah_{\mathbf{e}}(X)) = \Pr(X \mid \pi(X))$. The evidence vertices in $\pi(X)$ take configurations fixed by $\mathbf{e}$, that is $\Pr(X \mid \pi(X)) = \Pr(X \mid \pi(X) \setminus \mathbf{E}, e_1, \ldots, e_m)$ for all $e_i \in \pi(X) \cap \mathbf{E}$.*

Proof. See [Cheng and Druzdzel, 2000a]. □

Hence, to compute the posterior probability of a vertex that is not an ancestor of an evidence vertex, there is no need to change the parents of the vertex or its CPT. For vertices that are ancestors of evidence vertices, we use Bayes' formula and d-separation to explore the effects of evidence on those vertices. Without loss of generality, only one evidence vertex is considered. The result applies to an evidence vertex set by transitivity.

Lemma 1 *Let $BN_{\mathbf{e}}(G_{\mathbf{e}}, \Pr_{\mathbf{e}})$ be the* posterior *of a $BN = (G, \Pr)$ given evidence $\mathbf{e} = \{e\}$ for $E \in \mathbf{V}$. Let $X \in An(E)$. Then, $\Pr_{\mathbf{e}}(X \mid Ah_{\mathbf{e}}(X)) = \frac{\Pr(e \mid X, Ah(X))}{\Pr(e \mid Ah(X))} \Pr(X \mid \pi(X))$.*

Proof. Because $Ah_{\mathbf{e}}(X) = Ah(X)$ by Assumption 1, we have $\Pr_{\mathbf{e}}(X \mid Ah_{\mathbf{e}}(X)) = \frac{\Pr_{\mathbf{e}}(X, Ah(X))}{\Pr_{\mathbf{e}}(Ah(X))} = \frac{\Pr(X, Ah(X) \mid e)}{\Pr(Ah(X) \mid e)} = \frac{\Pr(X, Ah(X), e)}{\Pr(Ah(X), e)} = \frac{\Pr(e \mid X, Ah(X)) \Pr(X, Ah(X))}{\Pr(e \mid Ah(X)) \Pr(Ah(X))} = \frac{\Pr(e \mid X, Ah(X))}{\Pr(e \mid Ah(X))} \Pr(X \mid \pi(X))$ by using Theorem 2. □

Theorem 2 and Lemma 1 show that if we have $\Pr(e \mid X, Ah(X))$ and $\Pr(e \mid Ah(X))$ for all $X \in An(E)$ we can derive $\Pr_{\mathbf{e}}(X \mid Ah_{\mathbf{e}}(X))$ to compute $\Pr_{\mathbf{e}}(\mathbf{V}) = \prod_{V \in An(E)} \Pr(V \mid \pi(V)) \prod_{V \notin An(E)} \Pr_{\mathbf{e}}(V \mid Ah_{\mathbf{e}}(V))$ for the optimal importance function. However, there are two problems to derive $\Pr_{\mathbf{e}}$. Firstly, $Ah(X)$ is too large to construct a *posterior* $\mathbf{BN}_{\mathbf{e}}$ for $\Pr_{\mathbf{e}}$ in practice. Secondly, instead of the exact $\Pr_{\mathbf{e}}(X \mid Ah_{\mathbf{e}}(X))$ we have an estimate by importance sampling.

The parent sets of $X \in An(E)$ can be minimized by exploiting d-separation. Let $\alpha_e(X) \subseteq X \cup Ah(X)$ denote the minimal vertex set that d-separates evidence E and $X \cup Ah(X)$, thus $\langle E, X \cup Ah(X) \mid \alpha_e(X) \rangle$. We will refer to $\alpha_e(X)$ as the "shield" of X given E. Let $\beta_e(X) \subseteq Ah(X)$ denote the minimal vertex set that d-separates evidence E and $Ah(X)$, thus $\langle E, Ah(X) \mid \beta_e(X) \rangle$. We explore the relationship between $\alpha_e(X)$ and $\beta_e(X)$ below.

Lemma 2 *Let $BN_{\mathbf{e}}(G_{\mathbf{e}}, \Pr_{\mathbf{e}})$ be the* posterior *of a $BN = (G, \Pr)$ given evidence $\mathbf{e} = \{e\}$ for $E \in \mathbf{V}$. Then, $\beta_e(X) \subseteq (\alpha_e(X) \setminus X) \cup \pi(X)$ for all $X \in An(E)$.*

Proof. See Appendix C. □

Therefore, we can approach the optimal importance function $\Pr_{\mathbf{e}}(X \mid Ah_{\mathbf{e}}(X))$ by estimation of $\Pr_{\mathbf{e}}(X \mid (\alpha_e(X) \setminus X) \cup \pi(X))$ from importance samples.

Input: Evidence $E \in \mathbf{V}$ and $X \in An(E)$
Output: The set $S = \alpha_e(X)$
Data: array A, queue Q
$A \leftarrow \text{topSort}_\delta(Ah(X))$;
$S \leftarrow \{X\}$;
for $i \leftarrow |A|$ **to** 1 **do**
  $Q \leftarrow \emptyset$;
  push(Q, $A[i]$);
  while $Q \neq \emptyset$ **do**
    $V \leftarrow \text{pop}(Q)$;
    if $V = E$ **then** $S \leftarrow S \cup \{A[i]\}$; **break**;
    if $V \notin S \wedge V \in An(E) \wedge X \notin An(V)$ **then**
      push(children(V));
    end
  end
end

Algorithm 1: Computing the Shield $\alpha_e(X)$

3 Refractor Importance Sampling

The RIS algorithm modifies the BN structure according to the shield $\alpha_e(X)$ for vertices $X \in \mathbf{An}(\mathbf{E})$ by expanding the parent set of X and adjusting its CPT accordingly. Visually in the graph, *RIS refracts arcs from the evidence vertices*, which inspired the choice of name for the method. The algorithms and general procedure of RIS are introduced in this section.

3.1 Computing the Shield

Alg. 1 computes $\alpha_e(X)$ in $O(|\mathbf{A}|)$ worst-case time, assuming $An(\cdot)$ is determined in unit time (e.g. using a lookup table). Function topSort_δ topologically sorts the set $Ah(X)$ by topological order δ over $\mathbf{V}$ of the BN. Note that the shield $\alpha_e(X)$ can be computed in advance for each $X \in \mathbf{V}$ given evidence nodes $\mathbf{E}$.

3.2 Refractor Procedure

Alg. 2 modifies the graphical structure of BN. The time complexity of this algorithm is $O(|\mathbf{V}||\mathbf{A}|)$ if $|\mathbf{E}| \ll |\mathbf{V}|$, otherwise it is $O(|\mathbf{V}|^2|\mathbf{A}|)$. The CPT of a vertex X is updated by populating the expanded entries $\alpha_e(X) \setminus \{X\}$ using sampling data (described in Section 3.4).

Fig. 1 shows an example refractored BN using Alg. 2. E is the evidence node. Here, $\alpha_e(\mathsf{C}) = \{\mathsf{A}\}$ and $\alpha_e(\mathsf{B}) = \{\mathsf{A}\}$. Arcs $\mathsf{A} \rightarrow \mathsf{B}$ and $\mathsf{A} \rightarrow \mathsf{C}$ are added. Note that arc $\mathsf{A} \rightarrow \mathsf{B}$ adjusts for the fact that the influence relationship between A and B has changed through evidence E. Arc $\mathsf{E} \rightarrow \mathsf{D}$ is no longer required and can be removed as in [van der Gaag, 1996].

Input: $BN = (G, \Pr)$, evidence $\mathbf{e}$ for $\mathbf{E} \subseteq \mathbf{V}$
Output: refractored $BN_\mathbf{e}$
foreach $E \in \mathbf{E}$ **do**
  foreach $X \in An(E)$ **do**
    expand $\pi_\mathbf{e}(X) = (\alpha_e(X) \setminus \{X\}) \cup \pi(X)$;
    update the CPT of X;
  end
end

Algorithm 2: Refractor Procedure

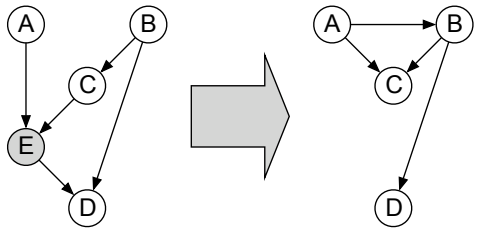


Figure 1: Refractor Example

3.3 General RIS Procedure

RIS utilizes both the qualitative and quantitative properties of a BN to approach the optimal importance function. The general procedure of RIS is:

1. The structure of the BN is modified by Algorithms 1 and 2. The CPTs of (a subset of) ancestor vertices of evidence vertices are expanded.
2. Update the CPT values through some specific learning algorithm (see Section 3.4 for details).
3. Sample the BN with an importance sampling algorithm using the new importance function.

3.4 Variations of RIS

Step 1 modifies the BN structure significantly, especially when the ancestor sets of evidence vertices are large, e.g. when evidence vertices are leafs. This increases the complexity of the BN. However, the effect of evidence on other vertices is attenuated when the path length between the evidence and the vertices is increased [Henrion, 1989]. Therefore, instead of modifying all ancestors $\mathbf{An}(\mathbf{E})$ of evidence $\mathbf{E}$ in Step 1, it is generally sufficient to select a subset of ancestors such as the combined parent set $\boldsymbol{\pi}(\mathbf{E})$.

Steps 1 and 2 are independent, because any importance function learning algorithm can be applied in Step 2. Steps 2 and 3 can be combined by using the same importance sampling algorithm for learning and inference. In our experiments, we used SIS and AIS-BN for both learning and inference (steps 2 and 3), referred to as RISSIS and RISAIS, respectively. AIS-BN will be referred to by AIS.

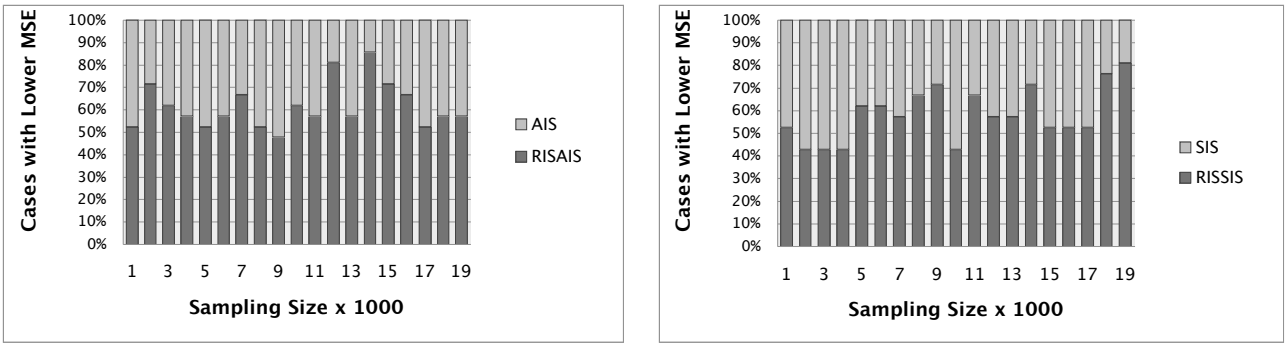


Figure 2: Synthetic BN Results: Ratio of Lowest MSE for RISAIS Versus AIS, and RISSIS Versus SIS.

4 Results

This section presents the experimental results of RISSIS and RISAIS compared to SIS and AIS for synthetic networks and two real-world networks.

4.1 Measurement

The *MSE* (mean squared error) metric was used to measure the error of the importance sampling results compared to the exact solution:

$$MSE = \sqrt{\frac{1}{\sum_{X_i \in \mathbf{X}} n_i} \sum_{X_i \in \mathbf{X}} \sum_{j=1}^{n_i} (\Pr'_{\mathbf{e}}(x_{ij}) - \Pr_{\mathbf{e}}(x_{ij}))^2} \ ,$$

where $\mathbf{X} = \mathbf{V} \setminus \mathbf{E}$. We also measured the KL-divergence of the approximate and exact posterior probability distributions:

$$\textit{KL-divergence} = \sum_{\textit{Cfg}(\mathbf{X})} \Pr_{\mathbf{e}}(\mathbf{x}) \ln \frac{\Pr_{\mathbf{e}}(\mathbf{x})}{\Pr'_{\mathbf{e}}(\mathbf{x})} \quad .$$

Recall that Theorem 1 gives a lower bound for the KL-divergence of the posterior probability distributions of SIS and AIS, which is indicated in the results by the *PostKLD* lower bound from Eq. (1).

The number of samples is taken as a measure of running time instead of CPU time in our experimental implementation. Recall that the overhead of RIS is fixed at startup when the evidence set can be predetermined. Furthermore, the RIS overhead is limited to collecting the updated CPT values during sampling (and learning in the case of RISAIS).

The reported sampling frequencies for AIS (and RISAIS) are for calculating the posterior results. Because AIS separates the importance function learning stage from the sampling stage, the actual number of samples taken for AIS (total sampling for importance function and sampling the results) is twice that of SIS. Recommended parameters [Cheng and Druzdzel, 2000a] are used in AIS and RISAIS.

4.2 Test Cases

Because computing the *MSE* is expensive and *PostKLD* is exponential in the number of vertices, small-sized synthetic BNs with random variables with two or three states and $|\mathbf{V}| = 20$ vertices and $|\mathbf{A}| = 30$ arcs were evaluated in our experiments. The CPT for each variable is randomly generated with uniform distribution for the probability interval $[0.1, 0.9]$ with bias for the extreme probabilities in intervals $(0, 0.1)$ and $(0.9, 1)$. For the experiments we generated 100 different synthetic BNs with these characteristics.

We also verified RIS with two real-world BNs: *Alarm-37* [Beinlich et al., 1989] and *HeparII-70* [Onisko, 2003]. The probability distributions of these networks are more extreme compared to the synthetic BNs. For each of the two BNs, 20 sets of evidence variables are randomly chosen, each with 10 evidence variables. For the *Alarm-37* and *HeparII-70* we choose to limit the refractoring to the parents nodes of the evidence set $\boldsymbol{\pi}(\mathbf{E})$ instead of $\mathbf{An}(\mathbf{E})$, see Section 3.4.

4.3 Results for Synthetic Test Cases

We compared the MSE of four algorithms, AIS, RISAIS, SIS, and RISSIS. For this comparison a selection of 21 BNs from the generated synthetic test case suite was made. The other 79 test cases have $\textit{PostKLD} \leq 0.1$, which means according to Theorem 1 that the RIS advantage is limited.

Fig. 2 shows the results for the 21 synthetic BNs, where the sample frequency is varied from 1,000 to 19,000 in increments of 1,000. The dark column in the figures represent the ratio of lowest MSE cases for RISAIS versus AIS and RISSIS versus SIS. A ratio of 50% or higher indicates that the RIS algorithm has lower error variance than the non-RIS algorithm. For RISAIS this is the case for all but one of the 19 measurements taken In total, the MSE is lowest for RISAIS in 61.4% on average over all samples. For RISSIS this is the case

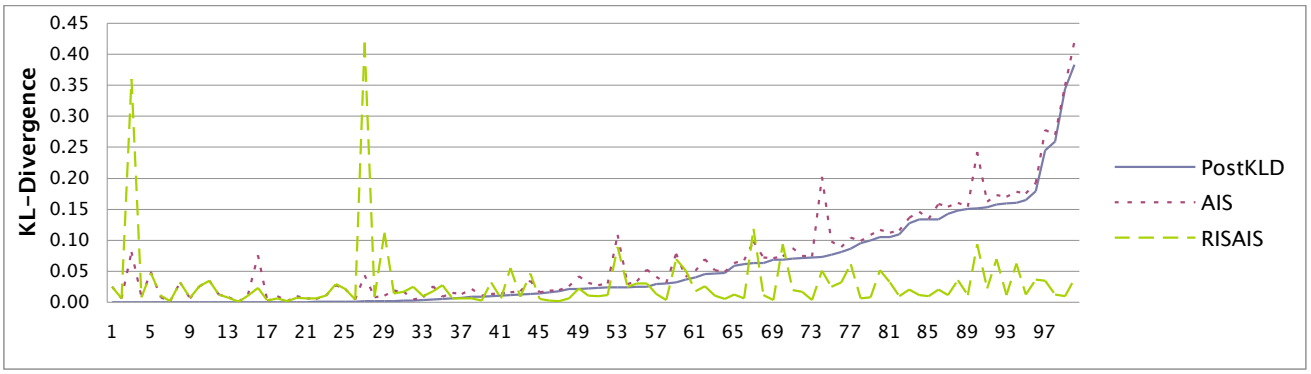


Figure 3: Synthetic BN Results: KL-Divergence of RISAIS and AIS with PostKLD Lower Bound

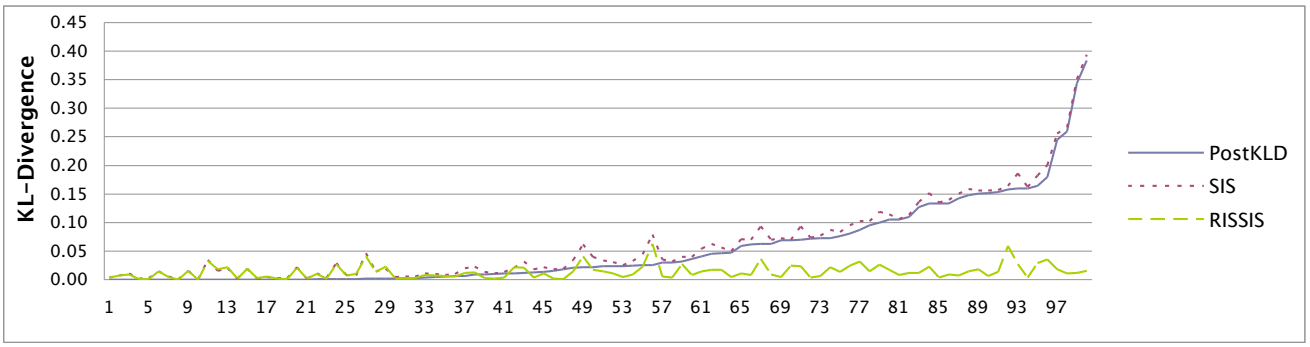


Figure 4: Synthetic BN Results: KL-Divergence of RISSIS and SIS with PostKLD Lower Bound

for all but four of the 19 measurements taken. In total, the MSE is lowest for RISSIS in 58.4% on average over all samples.

In fact, it is to be expected that the higher the *PostKLD* lower bound the better the RIS algorithms should perform. In order to determine the impact with increasing *PostKLD*, we selected all 100 synthetic BN test cases and measured the KL-divergence after 11,000 samples.

Fig. 3 shows the result for RISAIS, where the 100 BNs are ranked according to the *PostKLD*. Recall that the *PostKLD* is the lower bound on the KL-divergence of AIS. From the figure it can be concluded that AIS does not approach the exact solution for a significant number of test cases, whereas RISAIS is not limited by the bound due to the BN refractoring.

It should be noted that around points 1 and 26 in Fig. 3 the KL-divergence of RISAIS is worse compared to AIS. We believe the reason is that AIS heuristically changes the original CPT which has a negative impact on the RIS algorithm's ability to adjust the CPT to the optimal importance function.

Fig. 4 shows the result for RISSIS, where the 100 BNs are ranked according to the *PostKLD*. Interestingly, the RISSIS and SIS results are better on average than RISAIS and AIS. Note that the *PostKLD* lower bound is the same for AIS and SIS. However, in this study SIS appears to approach the *PostKLD* closer than AIS. Also here we can conclude that SIS does not approach the exact solution for a significant number of test cases, whereas RISSIS is not limited by the bound due to the BN refractoring.

4.4 Results for Alarm-37 and HeparII-70

Fig. 5 shows the results for *Alarm-37* and *HeparII-70*, where the sample frequency is varied from 1,000 to 19,000 in increments of 1,000. The dark column in the figures represent the ratio of lowest MSE cases for RISAIS versus AIS and RISSIS versus SIS. A ratio of 50% or higher indicates that the RIS algorithm has lower error variance than the non-RIS algorithm. For RISAIS this is the case for all but one of the 19 measurements taken. In total, the MSE is lowest for RISAIS in 56.7% on average over all samples. For RISSIS this is the case for all 19 measurements taken. In total, the MSE is lowest for RISSIS in 60.3% on average over all samples.

The combined results show that the RIS algorithms have reduced error variance for the synthetic networks

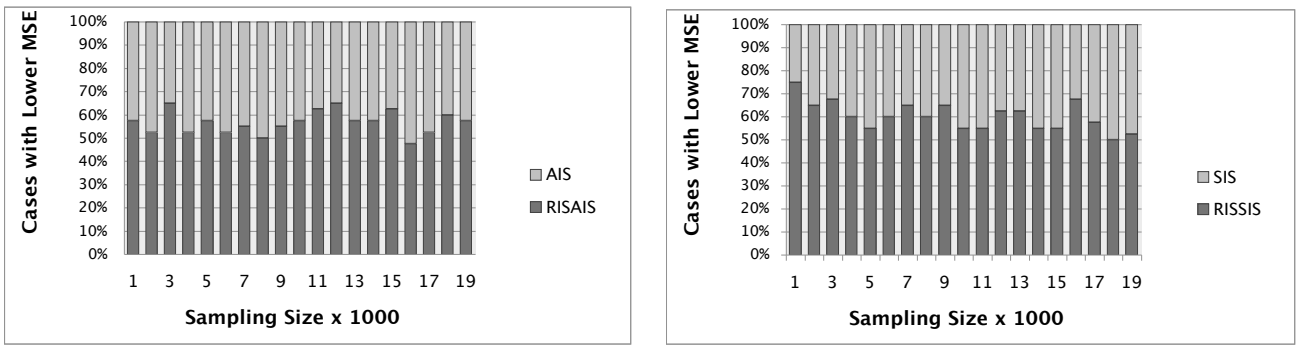


Figure 5: Ratio of Lowest MSE for RISAIS Versus AIS, and RISSIS Versus SIS (Alarm-37 and HeparII-70).

and the two real-world networks. The *PostKLD* lower bound limits the ability of AIS and SIS to approach the optimal importance function. By contrast, the RIS approach successfully eliminates this limitation of AIS and SIS, thereby providing an improvement to reduce error variance in BN importance sampling propagation under evidential reasoning.

5 Conclusions

In order to approach the optimal importance function for importance sampling propagation under evidential reasoning with a Bayesian network, a modification of the network's structure is necessary to eliminate the lower bound on the error variance. To this end, the proposed RIS algorithms refractor the network and adjust the conditional probability tables to minimize the divergence to the optimal importance function. The validity and performance of the RIS approach was empirically tested with a set of synthetic networks and two real-world networks.

Additional improvements of RIS are possible to achieve a better accuracy/cost ratio. The goal is to find an effective subset of the full shield size of an ancestor vertex of an evidence vertex or select a limited subset of the ancestors of evidence vertices that are refractored. Also some of the additionally introduced arcs could be removed with the arc removal algorithm [van Engelen, 1997] when they present a weak influence. Such strategies would reduce the complexity of the refractored network while still ensuring higher accuracy over current importance sampling algorithms.

References

[Beinlich et al., 1989] Beinlich, I., Suermondt, G., Chavez, R., and Cooper, G. (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the 2nd European Conference on AI and Medicine*, Berlin. Springer-Verlag.

[Bouckaert, 1994] Bouckaert, R. R. (1994). A stratified simulation scheme for inference in Bayesian belief networks. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 110–117.

[Chavez and Cooper, 1990] Chavez, R. M. and Cooper, G. F. (1990). A randomized approximation algorithm for probabilistic inference on Bayesian belief networks. *Networks*, 20:661–685.

[Cheng and Druzdzel, 2000a] Cheng, J. and Druzdzel, M. J. (2000a). AIS-BN: An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks. *Journal of Artificial Intelligence Research*, 13:155–188.

[Cheng and Druzdzel, 2000b] Cheng, J. and Druzdzel, M. J. (2000b). Computational investigations of low-discrepancy sequences in simulation algorithms for Bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 72–81. Morgan Kaufmann Publishers.

[Cheng and Druzdzel, 2000c] Cheng, J. and Druzdzel, M. J. (2000c). Latin hypercube sampling in Bayesian networks. In *Proceedings of the 13th International Florida Artificial Intelligence Research Symposium Conference (FLAIRS-2000)*, pages 287–292, Orlando, Florida. AAAI Publishers.

[Cooper, 1990] Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405.

[Dagum and Luby, 1993] Dagum, P. and Luby, M. (1993). Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60:141–153.

[Fung and Chang, 1989] Fung, R. and Chang, K. C. (1989). Weighting and integrating evidence for stochastic simulation in Bayesian networks. In *Proceedings of the 5th Conference on Uncertainty in Artificial Intelligence*, pages 209–219, New York, N. Y. Elsevier Science Publishing Company, Inc.

[Garvey and Lesser, 1994] Garvey, A. J. and Lesser, V. R. (1994). A survey of research in deliberative real-time artificial intelligence. *Real-Time Systems*, 6(3):317–347.

[Geman and Geman, 1984] Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741.

[Gilks et al., 1996] Gilks, W. R., Richardson, S., and Spiegelhalter, D. J. (1996). *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London.

[Guo and Hsu, 2002] Guo, H. and Hsu, W. (2002). A survey on algorithms for real-time Bayesian network inference. In *In the joint AAAI-02/KDD-02/UAI-02 workshop on Real-Time Decision Support and Diagnosis Systems*, Edmonton, Alberta, Canada.

[Henrion, 1988] Henrion, M. (1988). Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In *Proceedings of the 2nd Conference on Uncertainty in Artificial Intelligence*, pages 149–163, New York. Elsevier Science.

[Henrion, 1989] Henrion, M. (1989). Some practical issues in constructing belief networks. In Kanal, L., Levitt, T., and Lemmer, J., editors, *Proceedings of the 3rd Conference on Uncertainty in Artificial Intelligence*, pages 161–173, North Holland. Elsevier Science.

[Kullback, 1959] Kullback, S. (1959). *Information Theory and Statistics*. John Wiley and Sons, New York.

[MacKay, 1998] MacKay, D. (1998). Introduction to Monte Carlo methods. In Jordan, M., editor, *Learning in Graphical Models*. MIT Press.

[Neapolitan, 1990] Neapolitan, R. E. (1990). *Probabilistic Reasoning in Expert Systems*. John Wiley and Sons, NY.

[Onisko, 2003] Onisko, A. (2003). *Probabilistic Causal Models in Medicine: Application to Diagnosis of Liver Disorders*. PhD thesis, Institute of Biocybernetics and Biomedical Engineering, Polish Academy of Science, Warsaw.

[Pearl, 1987] Pearl, J. (1987). Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 32(2):245–257.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, CA.

[Rubinstein, 1981] Rubinstein, R. Y. (1981). *Simulation and Monte Carlo Method*. John Wiley and Sons, Hoboken, NJ.

[Russell and Norvig, 1995] Russell, S. and Norvig, P. (1995). Artificial intelligence: A modern approach. In *Prentice Hall Series in Artificial Intelligence*. Prentice Hall.

[Santos et al., 1995] Santos, E. J., Shimony, S. E., Solomon, E., and Williams, E. (1995). On a distributed anytime architecture for probabilistic reasoning. Technical report, AFIT/EN/TR95-02, Department of Electrical and Computer Engineering, Air Force Institute of Technology.

[Shachter and Peot, 1990] Shachter, R. D. and Peot, M. A. (1990). Simulation approaches to general probabilistic inference on belief networks. In *Proceedings of the 5th Conference on Uncertainty in Artificial Intelligence*, volume 5.

[Shannon, 1956] Shannon, C. E. (1956). The bandwagon. *IRE Transactions on Information Theory*, IT-2:3.

[van der Gaag, 1996] van der Gaag, L. (1996). On evidence absorption for belief networks. *International Journal of Approximate Reasoning*, 15(3):265–286.

[van Engelen, 1997] van Engelen, R. (1997). Approximating Bayesian belief networks by arc removal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):916–920.

Appendix

A Proof of Theorem 1

Proof. We use the KL-divergence (Cross Entropy) [Kullback, 1959] to measure the difference between the *posterior* $BN_{\mathbf{e}}$ and $ESBN_{\mathbf{e}}$. The KL-divergence $= \mathbf{E}_1[\ln \frac{\Pr_1(\mathbf{V})}{\Pr_2(\mathbf{V})}] = \sum_{Cfg(\mathbf{V})} \Pr_1(\mathbf{v}) \ln \frac{\Pr_1(\mathbf{v})}{\Pr_2(\mathbf{v})}$. Hence, the KL-divergence between *posterior* $BN_{\mathbf{e}}$ and $ESBN_{\mathbf{e}}$ is $\sum_{Cfg(\mathbf{X})} \Pr_{\mathbf{e}}(\mathbf{x}) \ln \frac{\Pr_{\mathbf{e}}(\mathbf{x})}{\Pr'_{\mathbf{e}}(\mathbf{x})}$ where $\mathbf{X} = \mathbf{V} \setminus \mathbf{E}$. This is further simplified as follows

$$\sum_{Cfg(\mathbf{X})} \Pr_{\mathbf{e}}(\mathbf{x}) \ln \frac{\Pr_{\mathbf{e}}(\mathbf{x})}{\Pr'_{\mathbf{e}}(\mathbf{x})} =$$
$$\sum_{Cfg(\mathbf{X})} \Pr(\mathbf{x} \mid \mathbf{e}) \ln \frac{\Pr(\mathbf{x}|\mathbf{e})}{\Pr'_{\mathbf{e}}(\mathbf{x})} =$$
$$\sum_{Cfg(\mathbf{X})} \Pr(\mathbf{x} \mid \mathbf{e}) \ln \frac{\prod_{x_j \in \mathbf{x}} \Pr(x_j|\pi(x_j)) \prod_{e_i \in \mathbf{e}} \Pr(e_i|\pi(e_i))}{\Pr(\mathbf{e}) \prod_{x_j \in \mathbf{x}} \Pr'_{\mathbf{e}}(x_j|\pi_{\mathbf{e}}(x_j))}$$
$$= \sum_{Cfg(\mathbf{X})} \Pr(\mathbf{x} \mid \mathbf{e}) \ln \frac{\prod_{x_j \in \mathbf{x}} \Pr(x_j|\pi(x_j)) \prod_{e_i \in \mathbf{e}} \Pr(e_i|\pi(e_i))}{\prod_{x_j \in \mathbf{x}} \Pr'_{\mathbf{e}}(x_j|\pi_{\mathbf{e}}(x_j))}$$
$$+\ln \tfrac{1}{\Pr(\mathbf{e})} \sum_{Cfg(\mathbf{X})} \Pr(\mathbf{x} \mid \mathbf{e}) =$$
$$\sum_{Cfg(\mathbf{X})} \Pr(\mathbf{x} \mid \mathbf{e}) \ln \frac{\prod_{x_j \in \mathbf{x}} \Pr(x_j|\pi(x_j)) \prod_{e_i \in \mathbf{e}} \Pr(e_i|\pi(e_i))}{\prod_{x_j \in \mathbf{x}} \Pr'_{\mathbf{e}}(x_j|\pi_{\mathbf{e}}(x_j))}$$
$$-\ln \Pr(\mathbf{e}) = \sum_{Cfg(\mathbf{X})} \Pr(\mathbf{x} \mid \mathbf{e}) \sum_{x_j \in \mathbf{x}} \ln \frac{\Pr(x_j|\pi(x_j))}{\Pr'_{\mathbf{e}}(x_j|\pi_{\mathbf{e}}(x_j))}$$
$$+\sum_{Cfg(\mathbf{X})} \Pr(\mathbf{x} \mid \mathbf{e}) \ln \prod_{e_i \in \mathbf{e}} \Pr(e_i \mid \pi(e_i)) \quad -\ln \Pr(\mathbf{e})$$
$$= \sum_{X_j \in \mathbf{X}} \sum_{Cfg(X_j, \pi(X_j))} \ln \frac{\Pr(x_j|\pi(x_j))}{\Pr'_{\mathbf{e}}(x_j|\pi_{\mathbf{e}}(x_j))}$$
$$\sum_{Cfg(\mathbf{X} \setminus \{X_j, \pi(X_j)\})} \Pr(\mathbf{x} \mid \mathbf{e}) \quad +\sum_{Cfg(\boldsymbol{\pi}(\mathbf{E}))}$$
$$\Pr(\boldsymbol{\pi}(\mathbf{e}) \mid \mathbf{e}) \ln \prod_{e_i \in \mathbf{e}} \Pr(e_i \mid \pi(e_i)) \quad -\ln \Pr(\mathbf{e}) =$$
$$\sum_{X_j \in \mathbf{X}} \sum_{Cfg(X_j, \pi(X_j))} \Pr(x_j, \pi(x_j) \mid \mathbf{e}) \ln \frac{\Pr(x_j|\pi(x_j))}{\Pr'_{\mathbf{e}}(x_j|\pi_{\mathbf{e}}(x_j))}$$
$$+\sum_{Cfg(\boldsymbol{\pi}(\mathbf{E}))} \Pr(\boldsymbol{\pi}(\mathbf{e}) \mid \mathbf{e}) \ln \prod_{e_i \in \mathbf{e}} \Pr(e_i \mid \pi(e_i))$$
$$-\ln \Pr(\mathbf{e}) = \sum_{X_j \in \mathbf{X}} \sum_{Cfg(X_j, \pi(X_j))} \Pr(x_j, \pi(x_j) \mid \mathbf{e})$$
$$\ln \Pr(x_j \mid \pi(x_j)) \quad +\sum_{X_j \in \mathbf{X}} \sum_{Cfg(X_j, \pi(X_j))}$$
$$\Pr(x_j, \pi(x_j) \mid \mathbf{e}) \ln \tfrac{1}{\Pr'_{\mathbf{e}}(x_j|\pi_{\mathbf{e}}(x_j))}$$
$$+\sum_{Cfg(\boldsymbol{\pi}(\mathbf{E}))} \Pr(\boldsymbol{\pi}(\mathbf{e}) \mid \mathbf{e}) \ln \prod_{e_i \in \mathbf{e}} \Pr(e_i \mid \pi(e_i))$$
$$-\ln \Pr(\mathbf{e}) \qquad \text{(Eq. 1)}$$

The first, third, and fourth terms in Eq. 1 are decided by the original probability distribution, so $\sum_{X_j \in \mathbf{X}} \sum_{Cfg(X_j, \pi(X_j))} \Pr(x_j, \pi(x_j) \mid \mathbf{e}) \ln \Pr(x_j \mid \pi(x_j))$ $+\sum_{Cfg(\boldsymbol{\pi}(\mathbf{E}))} \Pr(\boldsymbol{\pi}(\mathbf{e}) \mid \mathbf{e}) \ln \prod_{e_i \in \mathbf{e}} \Pr(e_i \mid \pi(e_i))$ $-\ln \Pr(\mathbf{e})$ is constant. To minimize the difference between *posterior* $BN_{\mathbf{e}}$ and $ESBN_{\mathbf{e}}$ the only choice is to minimize the following term:

$$\sum_{X_j \in \mathbf{X}} \sum_{Cfg(X_j, \pi(X_j))} \Pr(x_j, \pi(x_j) \mid \mathbf{e}) \ln \tfrac{1}{\Pr'_{\mathbf{e}}(x_j|\pi_{\mathbf{e}}(x_j))}$$
$$= \sum_{X_j \in \mathbf{X}} \sum_{Cfg(\pi(X_j))} \sum_{Cfg(X_j)} \Pr(x_j, \pi(x_j) \mid \mathbf{e})$$
$$\ln \tfrac{1}{\Pr'_{\mathbf{e}}(x_j|\pi_{\mathbf{e}}(x_j))}$$

This is equal to minimizing the term for each $X_j \in \mathbf{X}$ and each possible configuration of $\pi(x_j)$. $\sum_{Cfg(X_j)} \Pr(x_j, \pi(x_j) \mid \mathbf{e}) \ln \frac{1}{\Pr'_{\mathbf{e}}(x_j|\pi_{\mathbf{e}}(x_j))} =$ $\Pr(\pi(x_j) \mid \mathbf{e}) \sum_{Cfg(X_j)} \Pr(x_j \mid \pi(x_j), \mathbf{e}) \ln \frac{1}{\Pr'_{\mathbf{e}}(x_j|\pi(x_j)\setminus\mathbf{e})}$. We have $\sum_{Cfg(X_j)} \Pr'_{\mathbf{e}}(x_j \mid \pi_{\mathbf{e}}(x_j)) = \sum_{Cfg(X_j)} \Pr'_{\mathbf{e}}(x_j \mid \pi(x_j) \setminus \mathbf{e}) = 1$. According to Shannon's information theory [Shannon, 1956], to minimize $\sum_{Cfg(X_j)} \Pr(x_j \mid \pi(x_j), \mathbf{e}) \ln \frac{1}{\Pr'_{\mathbf{e}}(x_j|\pi(x_j)\setminus\mathbf{e})}$ we should set $\Pr'_{\mathbf{e}}(x_j \mid \pi(x_j) \setminus \mathbf{e}) = \Pr(x_j \mid \pi(x_j), \mathbf{e})$. This proves the Theorem 1. □

B Proof of Corollary 1

Proof. Let $\mathbf{X} = \mathbf{V} \setminus \mathbf{E}$, then $\sum_{Cfg(\mathbf{V}\setminus\mathbf{E})} \Pr_{\mathbf{e}}(\mathbf{v} \setminus \mathbf{e}) \ln \frac{\Pr_{\mathbf{e}}(\mathbf{v}\setminus\mathbf{e})}{\Pr'_{\mathbf{e}}(\mathbf{v}\setminus\mathbf{e})} =$ $\sum_{Cfg(\mathbf{X})} \Pr(\mathbf{x} \mid \mathbf{e}) \ln \frac{\Pr(\mathbf{x}|\mathbf{e})}{\Pr'_{\mathbf{e}}(\mathbf{x})} =$ $\sum_{Cfg(\mathbf{X})} \Pr(\mathbf{x} \mid \mathbf{e}) \ln \frac{\prod_{x_j \in \mathbf{x}} \Pr(x_j|\pi(x_j)) \prod_{e_i \in \mathbf{e}} \Pr(e_i|\pi(e_i))}{\Pr(\mathbf{e}) \prod_{x_j \in \mathbf{x}} \Pr'_{\mathbf{e}}(x_j|\pi_{\mathbf{e}}(x_j))}$. Since $\boldsymbol{\pi}(\mathbf{E}) \cap (\mathbf{V} \setminus \mathbf{E}) = \emptyset \Rightarrow \forall X_j \in \mathbf{X}, \Pr(x_j \mid \pi(x_j), \mathbf{e}) = \Pr(x_j \mid \pi(x_j))$, from Theorem 1, set $\Pr'_{\mathbf{e}}(x_j \mid \pi_{\mathbf{e}}(x_j)) = \Pr(x_j \mid \pi(x_j))$ to minimize the divergence, then $\sum_{Cfg(\mathbf{X})} \Pr(\mathbf{x} \mid \mathbf{e}) \ln \frac{\prod_{x_j \in \mathbf{x}} \Pr(x_j|\pi(x_j)) \prod_{e_i \in \mathbf{e}} \Pr(e_i|\pi(e_i))}{\Pr(\mathbf{e}) \prod_{x_j \in \mathbf{x}} \Pr'_{\mathbf{e}}(x_j|\pi_{\mathbf{e}}(x_j))} =$ $\sum_{Cfg(\mathbf{X})} \Pr(\mathbf{x} \mid \mathbf{e}) \ln \frac{\prod_{e_i \in \mathbf{e}} \Pr(e_i|\pi(e_i))}{\Pr(\mathbf{e})}$. Also from $\boldsymbol{\pi}(\mathbf{E}) \cap (\mathbf{V} \setminus \mathbf{E}) = \emptyset$, $\forall E_i \in \mathbf{E}$, $\pi(E_i) \subseteq \mathbf{E} \Rightarrow \prod_{e_i \in \mathbf{e}} \Pr(e_i \mid \pi(e_i)) = \Pr(\mathbf{e})$, so $\sum_{Cfg(\mathbf{X})} \Pr(\mathbf{x} \mid \mathbf{e}) \ln \frac{\prod_{e_i \in \mathbf{e}} \Pr(e_i|\pi(e_i))}{\Pr(\mathbf{e})} = 0$. The KL-divergence between $\Pr_{\mathbf{e}}$ and $\Pr'_{\mathbf{e}}$ is zero, thus $\Pr_{\mathbf{e}}(\mathbf{v} \setminus \mathbf{e}) = \Pr'_{\mathbf{e}}(\mathbf{v} \setminus \mathbf{e})$ according to [Kullback, 1959]. □

C Proof of Lemma 2

Proof. $\forall X_k \in Ah(X_j) \setminus \beta_e(X_j)$, consider the following three cases.

Case 1: If a path $X_k \to E$ exists then we show that this path is d-separated by $\alpha_e(X_j)$. There are two possibilities. First, $X_k \to E$ bypasses X_j, so it must pass one of the parents of X_j. Then $\pi(X_j)$ d-separates the path. Second, $X_k \to E$ does not pass X_j. Then the path must be d-separated by $\alpha_e(X_j) \setminus X_j$, so $(\alpha_e(X_j) \setminus X_j) \cup \pi(X_j)$ d-separates the path.

Case 2: If paths $N \to X_k$ and $N \to E$ exist, so $N \in Ah(X_j)$, and N d-separate the X_k and E, according to Case 1, $(\alpha_e(X_j) \setminus X_j) \cup \pi(X_j)$ d-separates path $N \to E$.

Case 3: If paths $X_k \to B$ and $E \to B$ exist, according to topological order $\{B, descendants\ of\ B\} \cap ((\alpha_e(X_j) \setminus X_j) \cup \pi(X_j)) = \emptyset$, so $(\alpha_e(X_j) \setminus X_j) \cup \pi(X_j)$ d-separates this path.

From cases 1 to 3 we see that $\langle E, Ah(X_j) \mid ((\alpha_e(X_j) \setminus X_j) \cup \pi(X_j)) \rangle$, so $\beta_e(X_j) \subseteq (\alpha_e(X_j) \setminus X_j) \cup \pi(X_j)$. □