MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY
and
CENTER FOR BIOLOGICAL AND COMPUTATIONAL LEARNING
DEPARTMENT OF BRAIN AND COGNITIVE SCIENCES

A.I. Memo No. 1565 February 2, 1996
C.B.C.L. Memo No. 132

Probabilistic Independence Networks for Hidden Markov Probability Models

Padhraic Smyth, David Heckerman, and Michael Jordan

Abstract

Graphical techniques for modeling the dependencies of random variables have been explored in a variety of different areas including statistics, statistical physics, artificial intelligence, speech recognition, image processing, and genetics. Formalisms for manipulating these models have been developed relatively independently in these research communities. In this paper we explore hidden Markov models (HMMs) and related structures within the general framework of probabilistic independence networks (PINs). The paper contains a self-contained review of the basic principles of PINs. It is shown that the well-known forward-backward (F-B) and Viterbi algorithms for HMMs are special cases of more general inference algorithms for arbitrary PINs. Furthermore, the existence of inference and estimation algorithms for more general graphical models provides a set of analysis tools for HMM practitioners who wish to explore a richer class of HMM structures. Examples of relatively complex models to handle sensor fusion and coarticulation in speech recognition are introduced and treated within the graphical model framework to illustrate the advantages of the general approach.

This report describes research done at the Department of Information and Computer Science, University of California, Irvine, the Jet Propulsion Laboratory, California Institute of Technology, Microsoft Research, the Center for Biological and Computational Learning, and the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. The authors can be contacted as pjs@aig.jpl.nasa.gov, heckerma@microsoft.com, and jordan@psyche.mit.edu. Support for CBCL is provided in part by a grant from the NSF (ASC–9217041). Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Dept. of Defense. MIJ gratefully acknowledges discussions with Steffen Lauritzen on the application of the IPF algorithm to UPINs.

1 Introduction

For multivariate statistical modeling applications, such as hidden Markov modeling for speech recognition, the identification and manipulation of relevant conditional independence assumptions can be a useful tool for model-building and analysis. There has recently been a considerable amount of work exploring the relationships between conditional independence in probability models and structural properties of related graphs. In particular, the *separation* properties of a graph can be directly related to *conditional independence* properties in a set of associated probability models.

The key point of this paper is that the analysis and manipulation of HMMs can be facilitated by exploiting the relationship between probability models and graphs. The major advantages to be gained are in:

- *Model Description:* A graphical model provides a natural and intuitive medium for displaying dependencies which exist between random variables. In particular, the *structure* of the graphical model clarifies the conditional independencies in the associated probability models, allowing model assessment and revision.

- *Computational Efficiency:* The graphical model is a powerful basis for specifying efficient algorithms for computing quantities of interest in the probability model, e.g., calculation of the probability of observed data given the model. These inference algorithms can be specified automatically once the initial structure of the graph is determined.

We will refer to both probability models and graphical models. Each consists of *structure* and *parameters*. The structure of the model consists of the specification of a set of *conditional independence relations* for the probability model, or a set of *(missing) edges* in the graph for the graphical model. The parameters of both the probability and graphical models consist of the specification of the joint probability distribution: in factored form for the probability model and defined locally on the nodes of the graph in the graphical model. The *inference* problem is that of the calculation of posterior probabilities of variables of interest given observable data and given a specification of the probabilistic model. The related task of *MAP identification* is the determination of the most likely state of a set of unobserved variables, given observed variables and the probabilistic model. The *learning* or *estimation* problem is that of determining the parameters (and possibly structure) of the probabilistic model from data.

This paper reviews the applicability and utility of graphical modeling to HMMs. Section 2 introduces the basic notation for probability models and associated graph structures. Section 3 summarizes relevant results from the literature on probabilistic independence networks (or PINs for short), in particular, the relationships which exist between separation in a graph and conditional independence in a probability model. Section 4 interprets the standard first-order HMM in terms of PINs. In Section 5 the standard algorithm for inference in a directed PIN is discussed and applied to the standard HMM in Section 6. A result of interest is that the F-B and Viterbi algorithms are shown to be special cases of this inference algorithm. Section 7 shows that the inference algorithms for undirected PINs are essentially the same as those already discussed for directed PINs. Section 8 introduces more complex HMM structures for speech modeling and analyzes them using the graphical model framework. Section 9 reviews known estimation results for graphical models and discusses their potential implications for practical problems in the estimation of HMM structures, and Section 10 contains summary remarks.

2 Notation and Background

Let $\mathbf{U} = \{X_1, X_2, \ldots, X_N\}$ represent a set of discrete-valued random variables. For the purposes of this paper we restrict our attention to discrete-valued random variables, however, many of the results stated generalize directly to continuous and mixed sets of random variables (Lauritzen and Wermuth 1989; Whittaker 1990). Let lower case x_1 denote one of the values of variable X_1: the notation $\sum_{x_1}$ is taken to mean the sum over all possible values of X_1. Let $p(x_i)$ be shorthand for the particular probability $p(X_i = x_i)$, whereas $p(X_i)$ represents the probability function for X_i (a table of values, since X_i is assumed discrete), $1 \leq i \leq N$. The full joint distribution function is $p(\mathbf{U}) = (X_1, X_2, \ldots, X_N)$ and $p(\mathbf{u}) = (x_1, x_2, \ldots, x_N)$ denotes a particular value assignment for $\mathbf{U}$.

If A, B and C are disjoint sets of random variables, the conditional independence relation $A \perp B|C$ is defined such that that A is independent of B given C, i.e., $p(A, B|C) = p(A|C)p(B|C)$. Conditional independence is symmetric. Note also that marginal independence (no conditioning) does not in general imply conditional independence, nor does conditional independence in general imply marginal independence (Whittaker 1990).

With any set of random variables $\mathbf{U}$ we can associate a graph G defined as $G = (V, E)$. V denotes the set of vertices or nodes of the graph such that there is a one-to-one mapping between the nodes in the graph and the random variables, i.e., $V = \{X_1, X_2, \ldots, X_N\}$. E denotes the set of edges, $\{e(i, j)\}$, where i and j are shorthand for the nodes X_i and X_j, $1 \leq i, j \leq N$. Edges of the form $e(i, i)$ are not of interest and thus are not allowed in the graphs discussed in this paper.

If the edges are ordered such that $e(i, j)$ means that the edge is directed from node i to node j, i is a *parent* of its *child* j. An *ancestor* of node i is a node which has as a child either i or another ancestor of i. A subset of nodes A is an *ancestral set* if it contains its own ancestors. A descendant of i is a either a child of i or a child of a descendant of i.

Two nodes i and j are *adjacent* in G if E contains the edge $e(i, j)$. A *path* is a sequence of distinct nodes $\{1, \ldots, m\}$ such that there exists an edge for each pair of nodes $\{l, l+1\}$ on the path. A graph is *singly-connected* if there exists only one path between any two nodes in the graph. A cycle is a path such the beginning and ending nodes on the path are the same. A directed cycle is a cycle of directed edges which all point in the same direction.

If E contains only undirected edges then the graph G is an *undirected graph (UG)*. If E contains only directed edges and no directed cycles, then G is an *acyclic directed graph (ADG)*. If E contains a mixture of directed and undirected edges, then it is referred to as a *mixed or chain graph*. We note in passing that there exists a theory for graphical independence models involving mixed graphs (Whittaker 1990) but mixed graphs will not be discussed further in this paper.

For an UG G, a subset of nodes C *separates* two other subsets of nodes A and B if every path joining every pair of nodes $i \in A$ and $j \in B$ contains at least one node from C. For ADGs and mixed graphs analagous, but somewhat more complicated, separation properties exist.

A cycle is *chordless* if no other than successive pairs of nodes in the cycle are adjacent. A graph G is *triangulated* if and only if the only chordless cycles in the graph contain no more than three nodes. Thus, if one can find a chordless cycle of length four or more, G is not triangulated.

A graph G is *complete* if there are edges between all pairs of nodes. The *cliques* of G are the largest subgraphs of G which are complete. A *clique tree* for G is a tree of cliques such that

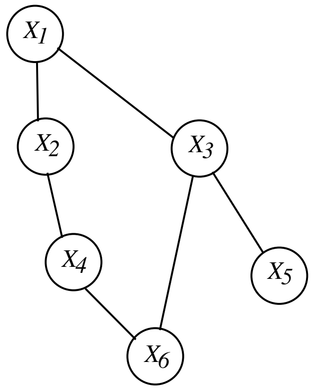

Figure 1: An example of a UPIN structure G which captures a particular set of conditional independence relationships among the set of variables $\{X_1, \ldots, X_6\}$. For example, $X_5 \perp \{X_1, X_2, X_4, X_6\} | \{X_3\}$.

there is a one-to-one node correspondence between the cliques of G and the nodes of the tree.

3 Probabilistic Independence Networks

We briefly review the relation between a probability model $p(\mathbf{U}) = p(X_1, \ldots, X_N)$ and a probabilistic independence network structure $G = (V, E)$. The results in this section are largely summarized versions of material in Pearl (1988) and Whittaker (1990) .

A probabilistic independence network structure (PIN structure) G, is a graphical statement of a set of conditional independence relations for a set of random variables $\mathbf{U}$. *Absence* of an edge $e(i, j)$ in G implies some independence relation between X_i and X_j. Thus, a PIN structure G is a particular way of specifying the independence relationships present in the probability model $p(\mathbf{U})$. We say that G implies a *set* of probability models $p(\mathbf{U})$, denoted as $\mathcal{P}_G$, i.e., $p(\mathbf{U}) \in \mathcal{P}_G$. In the reverse direction, a particular model $p(\mathbf{U})$ embodies a particular set of conditional independence assumptions which may or may not be representable in a consistent graphical form. One can derive all of the conditional independence properties and inference algorithms of interest for $\mathbf{U}$ without reference to graphical models. However, as has been emphasized in the statistical and AI literature, and is reiterated in this paper in the context of hidden Markov models, there are distinct advantages to be gained from using the graphical formalism.

3.1 Undirected Probabilistic Independence Networks (UPINs)

A UPIN is composed of both a UPIN structure and UPIN parameters. A UPIN *structure* specifies a set of conditional independence relations for a probability model in the form of an *undirected* graph. UPIN *parameters* consist of numerical specifications of a particular probability model consistent with the UPIN structure. Terms used in the literature to described UPINs of one form or another include Markov random fields, Markov networks, Boltzmann machines, and log-linear models.

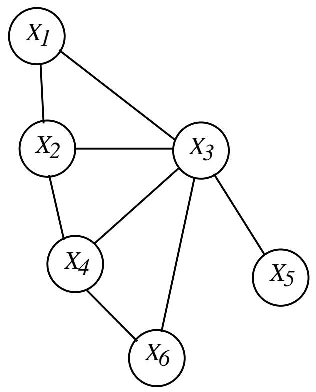

Figure 2: A triangulated version of the UPIN structure G from Figure 1.

3.1.1 Conditional Independence Semantics of UPIN Structures

Let A, B, and S be any disjoint subsets of nodes in an undirected graph (UG) G. G is an undirected probabilistic independence network structure (UPIN structure) for $p(\mathbf{U})$ if for any A, B, and S such that S separates A and B in G, the conditional independence relation $A \perp B|S$ holds in $p(\mathbf{U})$. The set of all conditional independence relations implied by separation in G constitute the (global) *Markov properties* of G. Figure 1 shows a simple example of a UPIN structure for 6 variables.

Thus, separation in the UPIN structure implies conditional independence in the probability model, i.e., it constrains $p(\mathbf{U})$ to belong to a set of probability models $\mathcal{P}_G$ which obey the Markov properties of the graph. Note that a complete UG is trivially a UPIN structure for any $p(\mathbf{U})$ in the sense that there are no constraints on $p(\mathbf{U})$. G is a *perfect undirected map* for p if G is a UPIN structure for p and all the conditional independence relations present in p are represented by separation in G. For many probability models p there are no perfect undirected maps. A weaker condition is that a UPIN structure G is *minimal* for a probability model $p(\mathbf{U})$ if the removal of any edge from G implies an independence relation which is not present in the model $p(\mathbf{U})$, i.e., the structure without the edge is no longer a UPIN structure for $p(\mathbf{U})$. Minimality is not equivalent to perfection (for UPIN structures) since, for example, there exist probability models with independencies which can not be represented as UPINs except for the complete UPIN structure. For example, if X and Y are marginally independent, but conditionally dependent given Z (see Figure 4(a) for an example), then the complete graph is the minimal UPIN structure for $\{X, Y, Z\}$ but it is not perfect because of the presence of an edge between X and Y.

3.1.2 Probability Functions on UPIN structures

Given a UPIN structure G, the joint probability distribution for $\mathbf{U}$ can be expressed as a simple factorization:

$$p(\mathbf{u}) = p(x_1, \ldots, x_N) = \prod_{V_C} a_C(x_C) \tag{1}$$

where V_C is the set of cliques of G, x_C represents a value assignment for the variables in a particular clique C, and the $a_C(x_C)$ are non-negative clique functions. The clique functions represent the particular *parameters* associated with the UPIN structure. This corresponds directly to the standard definition of a *Markov random field* (Isham 1981). The clique functions reflect the relative "compatibility" of the value assignments in the clique.

A model p is said to be *decomposable* if it has a minimal UPIN structure G which is triangulated (Figure 2). A UPIN structure G is decomposable if G is triangulated. For the special case of decomposable models, G can be converted to a *junction tree*, which is a tree of cliques of G arranged such that the cliques satisfy the *running intersection property*, namely, that each node in G which appears in any two different cliques also appears in all cliques on the path between these two cliques. Associated with each edge in the junction tree is a *separator* S, such that S contains the variables in the intersection of the two cliques which it links. Given a junction tree representation, one can factorize $p(\mathbf{U})$ as the product of clique marginals over separator marginals (Pearl 1988):

$$p(\mathbf{u}) = \frac{\prod_{C \in V_C} p(x_C)}{\prod_{S \in V_S} p(x_S)} \tag{2}$$

where $p(x_C)$ and $p(x_S)$ are the marginal (joint) distributions for the variables in clique C and separator S respectively and V_C and V_S are the set of cliques and separators in the junction tree.

This product representation is central to the results in the rest of the paper. It is the basis of the fact that globally consistent probability calculations on $\mathbf{U}$ can be carried out in a purely local manner. The mechanics of these local calculations will be described later in the paper. At this point it is sufficient to note that the complexity of the local inference algorithms scales as the sum of the sizes of the state-spaces of the cliques. Thus, local clique updating can make probability calculations on $\mathbf{U}$ much more tractable than using "brute force" inference, if the model decomposes into relatively small cliques.

Many probability models of interest may not be decomposable. However, we can define a *decomposable cover* G' for p such that G' is a triangulated, but not necessarily minimal, UPIN structure for p. Since any UPIN G can be triangulated simply by addition of the appropriate edges, one can always identify at least one decomposable cover G'. However, a decomposable cover may not be minimal in that it can contain edges which obscure certain independencies in the model p: for example, the complete graph is a decomposable cover for *all* possible probability models p. For efficient inference, the goal is to find a decomposable cover G' such that G' contains as few extra edges as possible over the original UPIN structure G. Later we discuss a specific algorithm for finding decomposable covers for arbitrary PIN structures. All *singly-connected* UPIN structures imply probability models $\mathcal{P}_{\mathcal{G}}$ which are decomposable.

Note that, given a particular probability model p and a UPIN G for p, the process of adding extra edges to G to create a decomposable cover does not change the underlying probability model p, i.e., the added edges are a convenience for manipulating the graphical representation, but the underlying numerical probability specifications remain unchanged.

An important point is that decomposable covers have the running intersection property and thus can be factored as in Equation 2: thus local clique updating is also possible with non-decomposable models via this conversion. Once again, the complexity of such local inference scales with the sum of the size of state-spaces of the cliques in the decomposable cover.

In summary, any UPIN structure can be converted to a junction tree permitting inference calculations to be carried out purely locally on cliques.

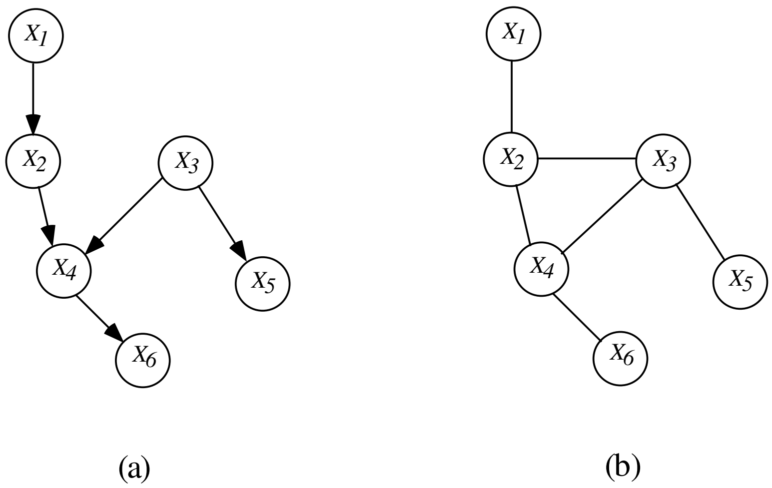

Figure 3: (a) A DPIN structure G^D which captures a set of independence relationships among the set $\{X_1, \ldots, X_5\}$. For example, $X_4 \perp X_1 | X_2$. (b) The moral graph G^M for G^D, where the parents of X_4 have been linked.

3.2 Directed Probabilistic Independence Networks (DPINs)

A DPIN is composed of both a DPIN structure and DPIN parameters. A DPIN *structure* specifies a set of conditional independence relations for a probability model in the form of a *directed* graph. DPIN *parameters* consist of numerical specifications of a particular probability model consistent with the DPIN structure. DPINs are referred to in the literature using different names, including Bayes network, belief network, recursive graphical model, causal (belief) network, and probabilistic (causal) network.

3.2.1 Conditional Independence Semantics of DPIN Structures

A DPIN structure is an ADG $G^D = (V, E)$ where there is a one-to-one correspondence between V and the elements of the set of random variables $\mathbf{U} = \{X_1, \ldots, X_N\}$.

The moral graph G^M of G^D is defined as the undirected graph obtained from G^D by placing undirected edges between all non-adjacent parents of each node and then dropping the directions from the remaining directed edges (see Figure 3b for an example). The term "moral" was coined to denote the "marrying" of "unmarried" (nonadjacent) parents.

Let A, B, and S be any disjoint subsets of nodes in G^D. G^D is a DPIN structure for $p(\mathbf{U})$ if for any A, B, and S such that S separates A and B in G^D, the conditional independence relation $A \perp B|S$ holds in $p(\mathbf{U})$. This is the same definition as for a UPIN structure except that separation has a different interpretation in the directed context: S separates A from B in a directed graph if S separates A from B in the moral (undirected) graph of the smallest ancestral set containing A, B, and S (Lauritzen et al. 1990). It can be shown that this is equivalent to the statement that a variable X_i is independent of all other nodes in the graph except for its descendants, given the values of its parents. Thus, as with a UPIN structure, the DPIN structure implies certain conditional independence relations, which in turn imply a set of probability models $p \in \mathcal{P}_{G^D}$. Figure 3a contains a simple example of a DPIN structure.

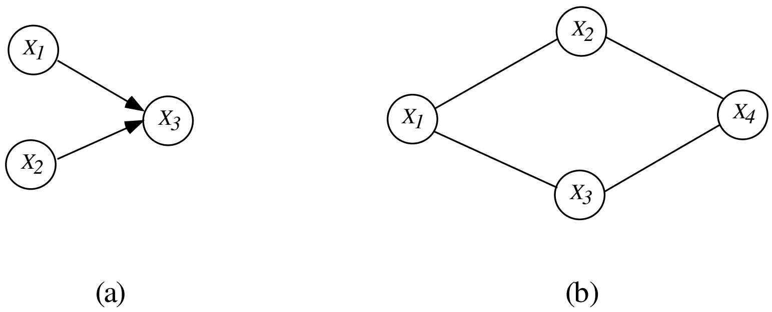

Figure 4: (a) The DPIN structure to encode the fact that X_3 depends on X_1 and X_2 but $X_1 \perp X_2$. For example, consider that X_1 and X_2 are two independent coin flips and that X_3 is a bell which rings when the flips are the same. There is no perfect UPIN structure which can encode these dependence relationships. (b) A UPIN structure which encodes $X_1 \perp X_4|\{X_2, X_3\}$ and $X_2 \perp X_3|\{X_1, X_4\}$. There is no perfect DPIN structure which can encode these dependencies.

3.2.2 Probability Functions on DPINs

A basic property of a DPIN structure is that it implies a direct factorization of the joint probability distribution $p(\mathbf{U})$:

$$p(\mathbf{u}) = \prod_{i=1}^{N} p(x_i|pa(x_i)) \tag{3}$$

where $pa(x_i)$ denotes a value assignment for the parents of X_i. A probability model p can be written in this factored form in a trivial manner by the conditioning rule. Consequently there are many possible DPIN structures consistent with a particular probability model p, potentially containing extra edges which hide true conditional independence relations. Thus, one can define *minimal DPIN structures* for p in a manner exactly equivalent to that of UPIN structures: deletion of an edge in a minimal DPIN structure G^D implies an independence relation which does not hold in $p \in \mathcal{P}_{G^D}$. Similarly, G^D is a *perfect DPIN structure* G for p if G^D is a DPIN structure for p and all the conditional independence relations present in p are represented by separation in G^D. As with UPIN structures, minimal does not imply perfect for DPIN structures. For example, consider the independence relations $X_1 \perp X_4|\{X_2, X_3\}$ and $X_2 \perp X_3|\{X_1, X_4\}$: the minimal DPIN structure contains an edge from X_3 to X_2 (see Figure 4(b)).

3.3 Differences between Directed and Undirected Graphical Representations

It is an important point that directed and undirected graphs possess different conditional independence semantics. There are common conditional independence relations which have perfect DPIN structures but no perfect UPIN structures and vice-versa (see Figure 4 for examples).

Does a DPIN structure have the same Markov properties as the UPIN structure obtained by dropping all the directions on the edges in the DPIN structure? The answer is yes if and only if the DPIN structure contains no subgraphs where a node has two or more non-adjacent parents (Whittaker 1990; Pearl et al. 1990). In general, it can be shown that if a UPIN structure G

for p is decomposable (triangulated) then it has the same Markov properties as some DPIN structure for p.

On a more practical level, DPIN structures are frequently used to encode causal information, i.e., to formally represent the belief that X_i preceeds X_j in some causal sense, e.g., temporally. DPINs have found application in causal modelling in applied statistics and artificial intelligence. Their popularity in these fields stems from the fact that the joint probability model can be specified directly via Equation 3, i.e., via the specification of conditional probability tables or functions (Spiegelhalter et al. 1991). In contrast, UPINs must be specified in terms of clique functions (as in Equation 1) which may not be as easy to work with (cf. Geman and Geman (1984), Modestino and Zhang (1992) and Vandermeulen et al. (1994) for examples of ad hoc design of clique functions in image analysis). UPINs are more frequently used in problems such as image analysis and statistical physics where associations are thought to be correlational rather than causal.

3.4 From DPINs to (Decomposable) UPINs

The moral UPIN structure G^M (obtained from the DPIN structure G^D) does not imply any new independence relations which are not present in G^D. As with triangulation, however, the additional edges may obscure conditional independence relations which are implicit in the numeric specification of the original probability model p associated with the DPIN structure G^D. Furthermore, G^M may not be triangulated (decomposable). By the addition of appropriate edges, the moral graph can be converted to a (non-unique) triangulated graph G', namely a decomposable cover for G^M. In this manner, for any probability model p for which G^D is a DPIN structure, one can construct a decomposable cover G' for p.

This mapping from DPIN structures to UPIN structures was first discussed in the context of efficient inference algorithms by Lauritzen and Spiegelhalter (1988). The advantage of this mapping derives from the fact that analysis and manipulation of the resulting UPIN is considerably more direct than dealing with the original DPIN. Furthermore, it has been shown that many of the inference algorithms for DPINs are in fact special cases of inference algorithms for UPINs and can be considerably less efficient (Shachter et al. 1994).

4 Modeling HMMs as PINs

4.1 PINs for HMMs

In hidden Markov modeling problems (Poritz 1988; Rabiner 1989) we are interested in the set of random variables $\mathbf{U} = \{H_1, O_1, H_2, O_2, \ldots, H_{N-1}, O_{N-1}, H_N, O_N\}$, where H_i is a discrete-valued hidden variable at index i, and O_i is the corresponding discrete-valued observed variable at index i, $1 \leq i \leq N$ (the results here can be directly extended to continuous-valued observables). The index i denotes a sequence from 1 to N, for example, discrete time steps. Note that O_i is considered univariate for convenience: the extension to the multivariate case with d observables is straightforward but is omitted here for simplicity since it does not illuminate the conditional independence relationships in the HMM.

The well-known simple first-order HMM obeys the following two conditional independence relations:

$$H_i \perp \{H_1, O_1, \ldots, H_{i-2}, O_{i-2}, O_{i-1}\} | H_{i-1}, \qquad 2 \leq i \leq N \tag{4}$$

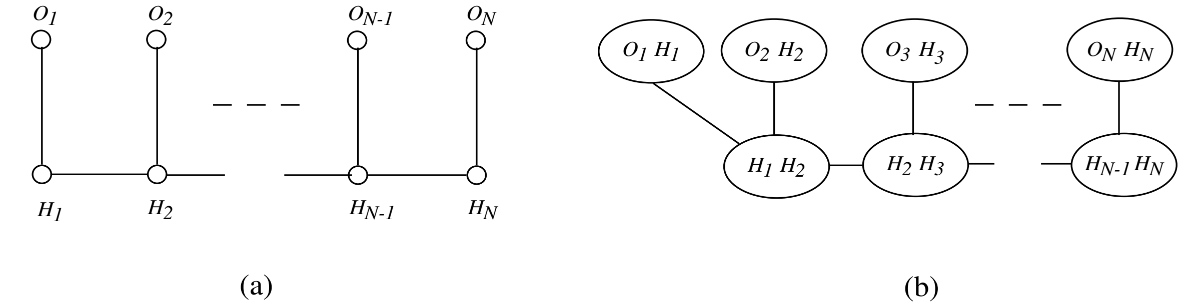

Figure 5: (a) The PIN structure for HMM(1,1) (b) A corresponding junction tree.

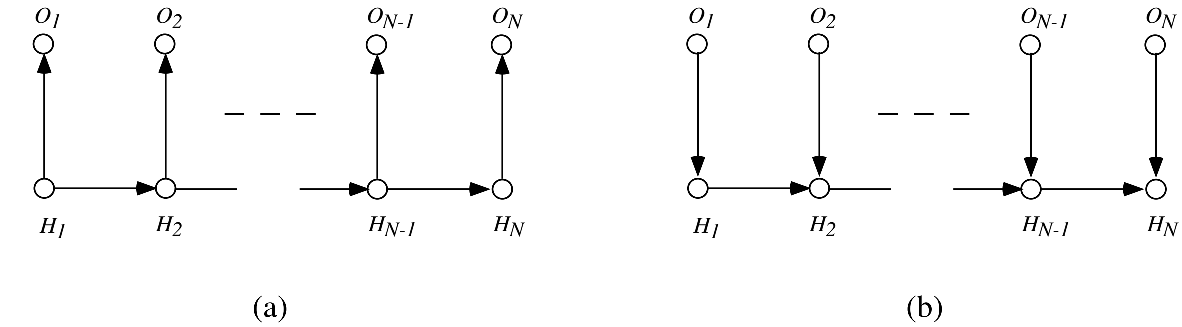

Figure 6: DPIN structures for HMM(1,1): (a) the DPIN structure for the HMM(1,1) probability model, (b) a DPIN structure which is not a DPIN structure for the HMM(1,1) probability model.

and

$$O_i \perp \{H_1, O_1, \ldots, H_{i-1}, O_{i-1}\} | H_i, \qquad 2 \leq i \leq N \tag{5}$$

We will refer to this "first-order" hidden Markov probability model as HMM(1,1): the notation HMM(K, J) is defined such that the model has state memory of depth K and contains J separate underlying state processes. The notation will be clearer in later sections when we discuss specific examples with $K, J > 1$.

Construction of a PIN for HMM(1,1) is particularly simple. In the undirected case, assumption 1 requires that each state H_i is only connected to H_{i-1} from the set $\{H_1, O_1, \ldots, H_{i-2}, O_{i-2}, O_{i-1}\}$. Assumption 2 requires that O_i is only connected to H_i. The resulting UPIN structure for HMM(1,1) is shown in Figure 5a. This graph is singly-connected and thus implies a decomposable probability model p for HMM(1,1), where the cliques are of the form $\{H_i, O_i\}$ and $\{H_{i-1}, H_i\}$ (Figure 5b). In Section 5 we will see how the joint probability function can be expressed as a product function on the junction tree, thus leading to a junction tree definition of the familiar F-B and Viterbi inference algorithms.

For the directed case the connectivity for the DPIN structure is the same. It is natural to choose the directions on the edges between H_{i-1} and H_i as going from $i-1$ to i (although the reverse direction could also be chosen without changing the Markov properties of the graph). The directions on the edges between H_i and O_i must be chosen as going from H_i to O_i rather than in the reverse direction (Figure 6a). In reverse (Figure 6b) the arrows would imply that O_i is marginally independent of H_{i-1} which is not true in the HMM(1,1) probability model. The proper direction for the edges implies the correct relation, namely that O_i is *conditionally independent* of H_{i-1} given H_i.

The DPIN structure for HMM(1,1) does not possess a subgraph with non-adjacent parents. As stated earlier this implies that the implied independence properties of the DPIN structure are the same as those of the corresponding UPIN structure obtained by dropping the directions from the edges in the DPIN structure, and thus they both result in the same junction tree structure (Figure 5b). Thus, for the HMM(1,1) probability model, the minimal directed and undirected graphs possess the same Markov properties, i.e., imply the same conditional independence relations. Furthermore, both PIN structures are perfect maps for the directed and undirected cases respectively.

4.2 Inference and MAP Problems in HMMs

In the context of HMMs, the most common inference problem is the calculation of the likelihood of the observed evidence given the model, i.e., $p(o_1, \ldots, o_N|\text{model})$, where the $o_1, \ldots, o_N$ denote observed values for $O_1, \ldots, O_N$. (In this section we will assume that we are dealing with one particular model where the structure and parameters have already been determined and, thus, we will not explicitly indicate conditioning on the model). The "brute force" method for obtaining this probability would be to sum out the unobserved state variables from the full joint probability distribution:

$$p(o_1, \ldots, o_N) = \sum_{h_1,\ldots,h_N} p(H_1, o_1, \ldots, H_N, o_N) \tag{6}$$

where h_i denotes the possible values of hidden variable H_i.

Another inference calculation of interest is the calculation of $p(h_i|o_1, \ldots, o_N)$, for any or all i, namely, the probability of a particular hidden state value given the observed evidence. Inferring the posterior state probabilities is useful when the states have direct physical interpretations (as in fault monitoring applictions (Smyth 1994)) and is also implicitly required during the standard Baum-Welch learning algorithm for HMM(1,1).

In general, both of these computations scale as m^N where m is the number of states for each hidden variable. In practice, the F-B algorithm (Poritz 1988; Rabiner 1989) can perform these inference calculations with much lower complexity, namely Nm^2. The likelihood of the observed evidence can be obtained with the forward step of the F-B algorithm: calculation of the state posterior probabilities requires both forward and backward steps. The F-B algorithm relies on a factorization of the joint probability function to obtain locally recursive methods One of the key points in this paper is that the graphical modeling approach provides an *automatic* method for determining such local efficient factorizations, for an arbitrary probabilistic model, *if efficient factorizations exist* given the CI relations specified in the model.

The MAP identification problem in the context of HMMs involves identifying the most likely hidden state sequence given the observed evidence. Just as with the inference problem, the Viterbi algorithm provides an efficient, locally recursive method for solving this problem with complexity Nm^2, and again, as with the inference problem, the graphical modeling approach provides an automatic technique for determining efficient solutions to the MAP problem for arbitrary models, if an efficient solution is possible given the structure of the model.

5 Inference and MAP Algorithms for DPINs

Inference and MAP algorithms for DPINs and UPINS are quite similar: the UPIN case involves some subtleties not encountered in DPINs and so discussion of UPIN inference and MAP algorithms is deferred until Section 7. The inference algorithm for DPINs (developed by Jensen, Lauritzen and Oleson (1990) and hereafter referred to as the JLO algorithm) is a descendant of an inference algorithm first described by Lauritzen and Spiegelhalter (1988). The JLO algorithm applies to discrete-valued variables: extensions to the JLO algorithm for Gaussian and Gaussian-mixture distributions are discussed in Lauritzen and Wermuth (1989). A closely related algorithm to the JLO algorithm, developed by Dawid (1992), solves the MAP identification problem with the same time-complexity as the JLO inference algorithm.

We show that the JLO and Dawid algorithms are strict generalizations of the well-known F-B and Viterbi algorithms for HMM(1,1), in that they can be applied to arbitrarily complex graph structures (and thus a large family of probabilistic models beyond HMM(1,1)) and handle missing values, partial inference, and so forth in a straightforward manner.

There are many variations on the basic JLO and Dawid algorithms. For example, Pearl (1988) describes related versions of these algorithms in his early work . However, it can be shown (Shachter et al. 1994) that all known exact algorithms for inference on DPINs are equivalent at some level to the JLO and Dawid algorithms. Thus, it is sufficient to consider the JLO and Dawid algorithms in our discussion as they subsume other graphical inference algorithms.

The JLO and Dawid algorithms operate as a two-step process:

1. The *construction* step: this involve a series of sub-steps where the original directed graph is moralized and triangulated, a junction tree is formed, and the junction tree is initialized.

2. The *propagation* step: the junction tree is used in a local message-passing manner to propagate the effects of observed evidence, i.e., to solve the inference and MAP problems.

The first step is carried out only once for a given graph. The second (propagation) step is carried out each time a new inference for the given graph is requested.

5.1 The Construction Step of the JLO Algorithm: From DPIN structures to Junction Trees

We illustrate the construction step of the JLO algorithm using the simple DPIN structure, G^D, over discrete variables $\mathbf{U} = \{X_1, \ldots, X_6\}$ shown in Figure 7a. The JLO algorithm first constructs the moral graph G^M (Figure 7b). It then triangulates the moral graph G^M to obtain a decomposable cover G' (Figure 7c). The algorithm operates in a simple greedy manner based on the fact that a graph is triangulated if and only if all of its nodes can be eliminated, where a node can be eliminated whenever all of its neighbors are pairwise linked. Whenever a node is eliminated, it and its neighbors define a clique in the junction tree that is eventually constructed. Thus, we can triangulate a graph and generate the cliques for the junction tree by eliminating nodes in some order, adding links if necessary. If no node can be eliminated without adding links, then we choose the node that can be eliminated by adding the links that yield the clique with the smallest state-space (Jensen 1995).

After triangulation the JLO algorithm constructs a junction tree from G', i.e., a clique tree satisfying the running intersection property. The junction tree construction is based on the

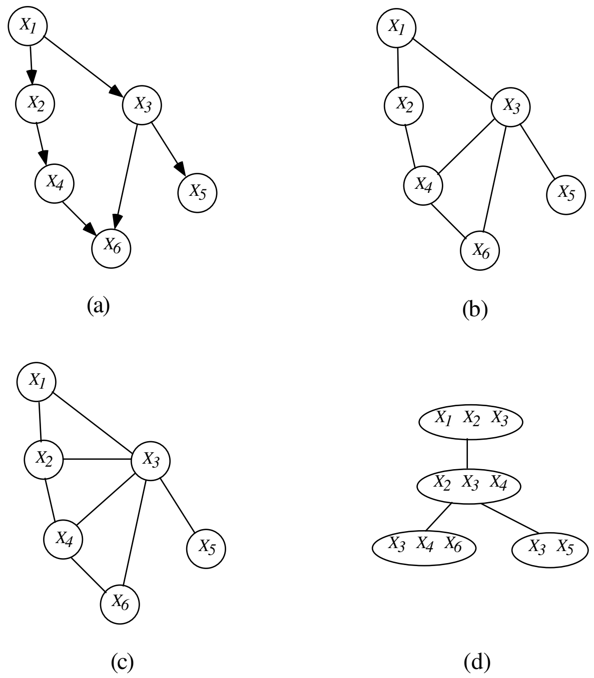

Figure 7: (a) A simple DPIN structure G^D. (b) The corresponding (undirected) moral graph G^M. (c) The corresponding triangulated graph G'. (d) The corresponding junction tree.

following fact. Define the weight of a link between two cliques as the number of variables in their intersection. Then, a tree of cliques will satisfy the running intersection property if and only if it is a spanning tree of maximal weight. Thus, the JLO algorithm constructs a junction tree by choosing successively a link of maximal weight unless it creates a cycle. The junction tree constructed from the cliques defined by the DPIN structure triangulation in Figure 7c is shown in Figure 7d.

The worst-case complexity is $O(N^3)$ for the triangulation heuristic and $O(N^2 \log N)$ for the maximal spanning tree portion of the algorithm. This construction step is carried out only once as an initial step to convert the original graph to a junction tree representation.

5.2 Initializing the Potential Functions in the Junction Tree

The next step is to take the numeric probability specifications as defined on the directed graph G^D (Equation 3) and convert this information into the general form for a junction tree representation of p (Equation 2). This is achieved by noting that each variable X_i is contained in at least one clique in the junction tree. Assign each X_i to just one such clique and for each clique define the potential function $a_C(C)$ to be either the product of $p(X_i|pa(X_i))$ or 1 if no variables are assigned to that clique. Define the separator potentials (in Equation 2) to be 1 initially.

In the section which follows we describe the general JLO algorithm for propagating messages through the junction tree to achieve globally consistent probability calculations. At this point it is sufficient to know that a schedule of local message passing can be defined which converges to a globally consistent *marginal* representation for p, i.e., the potential on any clique or separator is the marginal for that clique or separator (the joint probability function). Thus, via local message-passing, one can go from the initial potential representation defined above to a marginal representation:

$$p(\mathbf{u}) = \frac{\prod_{C \in V_C} p(x_C)}{\prod_{S \in V_S} p(x_S)} \tag{7}$$

At this point the junction tree is initialized. This operation in itself is not that useful, of more interest is the ability to propagate information through the graph given some observed data and the initialized junction tree, e.g., to calculate the posterior distributions of some variables of interest.

From this point onwards we will implicitly assume that the junction tree has been initialized as described above so that the potential functions are the local marginals.

5.3 Local Message Propagation in Junction Trees Using The JLO Algorithm

In general $p(\mathbf{U})$ can be expressed as

$$p(\mathbf{u}) = \frac{\prod_{C \in V_C} a_C(x_C)}{\prod_{S \in V_S} b_S(x_S)} \tag{8}$$

where the a_C and b_S are non-negative potential functions (the potential functions could be the initial marginals described above for example). $K = (\{a_C : C \in V_C\}, \{b_S : S \in S_C\})$ is a *representation* for $p(\mathbf{U})$. A factorizable function $p(\mathbf{U})$ can admit many different representations, i.e., many different sets of clique and separator functions which satisfy Equation 8 given a particular $p(\mathbf{U})$.

As mentioned above, the JLO algorithm carries out globally consistent probability calculations via local message-passing on the junction tree, i.e., probability information is passed between neighboring cliques and clique and separator potentials are updated based on this local information. A key point is that the cliques and separators are updated in a fashion which ensures that at all times K is a representation for $p(\mathbf{U})$, i.e., Equation 8 holds at all times. Eventually the propagation converges to the marginal representation given the initial model and the observed evidence.

The message-passing proceeds as follows. We can define a *flow* from clique C_i to C_j in the following manner where C_i and C_j are two cliques which are adjacent in the junction tree. Let S_k be the separator for these two cliques. Define

$$b^*_{S_k}(x_{S_k}) = \sum_{C_i \setminus S_k} a_{C_i}(x_{C_i}) \tag{9}$$

and

$$a^*_{C_j}(x_{C_j}) = a_{C_j}(x_{C_j}) \lambda_{S_k}(x_{S_k}) \tag{10}$$

where

$$\lambda_{S_k}(x_{S_k}) = \frac{b^*_{S_k}(x_{S_k})}{b_{S_k}(x_{S_k})}. \tag{11}$$

$\lambda_{S_k}(x_{S_k})$ is the *update factor*. Passage of a flow corresponds to updating the neighboring clique with the probability information contained in the originating clique. This flow induces a new representation $K^* = (\{a^*_C : C \in V_C\}, \{b^*_S : S \in S_C\})$ for $p(\mathbf{U})$.

A *schedule* of such flows can be defined such that all cliques are eventually updated with all relevant information and the junction tree reaches an *equilibrium* state. The most direct scheduling scheme is a two-phase operation where one node is denoted the *root* of the junction tree. The *collection* phase involves passing flows along all edges towards the root-clique (if a node is scheduled to have more than one incoming flow, the flows are absorbed sequentially). Once collection is complete, the *distribution* phase involves passing flows out from this root in the reverse direction along the same edges. There are at most two flows along any edge in the tree in a non-redundant schedule. Note that the directionality of the flows in the junction tree need have nothing to do with any directed edges in the original DPIN structure.

5.4 The JLO Algorithm for Inference given Observed Evidence

The particular case of calculating the effect of observed evidence (inference) is handled in the following manner. Consider that we observe evidence of the form $e = \{X_i = x_i^*, X_j = x_j^*, \ldots\}$ and $\mathbf{U}^e = \{X_i, X_j, \ldots\}$ denotes the set of variables which have been observed. Let $\mathbf{U}^h = \mathbf{U} \setminus \mathbf{U}^e$ denote the set of hidden or unobserved variables and $\mathbf{u}^h$ a value assignment for $\mathbf{U}^h$.

Consider the calculation of $p(\mathbf{U}^h|e)$. Define an evidence function $g^e(x_i)$ such that

$$g^e(x_i) = \begin{cases} 1 & \text{if } x_i = x_i^* \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

Let

$$f^*(\mathbf{u}) = p(\mathbf{u}) \prod_{\mathbf{U}^e} g^e(x_i) \tag{13}$$

Thus, we have that $f^*(\mathbf{u}) \propto p(\mathbf{u}^h|e)$. To obtain $f^*(\mathbf{u})$ by operations on the junction tree one proceeds as follows. First assign each observed variable $X_i \in \mathbf{U}^e$ to one particular clique which contains it (this is termed "entering the evidence into the clique"). Let C^E denote the set of all cliques into which evidence is entered in this manner. For each $C \in C^E$ let

$$g_C(x_C) = \prod_{\{i: X_i \text{ is entered into } C\}} g^e(x_i) \tag{14}$$

Thus,

$$f^*(\mathbf{u}) = p(\mathbf{u}) \times \prod_{C \in C^E} g_C(x_C). \tag{15}$$

One can now propagate the effects of these modifications throughout the tree using the collect and distribute schedule described in 5.3. Let x_C^h denote a value assignment of the hidden (unobserved) variables in clique C. When the schedule of flows is complete one gets a new representation K_f^* such that the local potential on each clique is $f^*(x_C) = p(x_C^h, e)$, i.e., the joint probability of the local unobserved clique variables and the observed evidence (Jensen et al. 1990) (similarly for the separator potential functions). If one marginalizes at the clique over the unobserved local clique variables,

$$\sum_{X_C^h} p(x_C^h, e) = p(e), \tag{16}$$

one gets the probability of the observed evidence directly. Similarly, if one normalizes the potential function at a clique to sum to 1, one obtains the conditional probability of the local unobserved clique variables given the evidence, $p(x_C^h|e)$.

5.5 Complexity of the Propagation Step of the JLO Algorithm

In general, the time complexity T of propagation within a junction tree is $O(\sum_{i=1}^{N_C} s(C_i))$ where N_C is the number of cliques in the junction tree and $s(C_i)$ is the number of states in the joint state-space of C_i (equal to the product over each variable in C_i of the number of states of each variable). Thus, for inference to be efficient, we need to construct junction trees with small clique sizes. Problems of finding optimally small junction trees (e.g., finding the junction tree with the smallest maximal clique) are NP-hard. Nonetheless, the heuristic algorithm for triangulation described earlier has been found to work well in practice (Jensen et al. 1990; Jensen 1995).

6 Inference and MAP Calculations in HMM(1,1)

6.1 The F-B Algorithm for HMM(1,1) is a Special Case of the JLO Algorithm

Figure 5b shows the junction tree for HMM(1,1). In this section we apply the JLO algorithm to the HMM(1,1) junction tree structure to obtain a particular inference algorithm for HMM(1,1). As mentioned earlier, the HMM(1,1) inference problem consists of being given a set of values for the observable variables,

$$e = \{O_1 = o_1, O_2 = o_2, \ldots, O_N = o_N\} \tag{17}$$

and inferring the likelihood of e given the model. As described in the previous section this problem can be solved exactly by local propagation in any junction tree using the JLO inference algorithm.

Let the final clique in the chain containing (H_{N-1}, H_N) be the root clique. Thus, a non-redundant schedule consists of first recursively passing flows from each (O_i, H_i) and (H_{i-2}, H_{i-1}) to each (H_{i-1}, H_i) in the appropriate sequence (the "collect" phase), and then distributing flows out in the reverse direction from the root clique. If we are only interested in calculating the likelihood of e given the model, then the distribute phase is not necessary since we can simply marginalize over the local variables in the root clique to obtain $p(e)$.

A comment on notation: subscripts on potential functions and update factors indicate which variables have been used in deriving that potential or update factor, e.g., f_{O_1} indicates that this potential has been updated based on information about O_1 but not using information about any other variables.

Assume that the junction tree has been initialized so that the potential function for each clique and separator is the local marginal. Given the observed evidence e, each individual piece of evidence $O = o_i^*$ is entered into its clique (O_i, H_i) such that each clique marginal becomes $f^*_{O_i}(h_i, o_i) = p(h_i, o_i^*)$ after entering the evidence (as in Equation 14).

Consider the portion of the junction tree in Figure 8, and in particular the flow between (O_i, H_i) and (H_{i-1}, H_i). By definition the potential on the separator H_i is updated to

$$f^*_{O_i}(h_i) = \sum_{o_i} f^*(h_i, o_i) = p(h_i, o_i^*) \tag{18}$$

The update factor from this separator flowing into clique (H_{i-1}, H_i) is then

$$\lambda_{O_i}(h_i) = \frac{p(h_i, o_i^*)}{p(h_i)} = p(o_i^* | h_i). \tag{19}$$

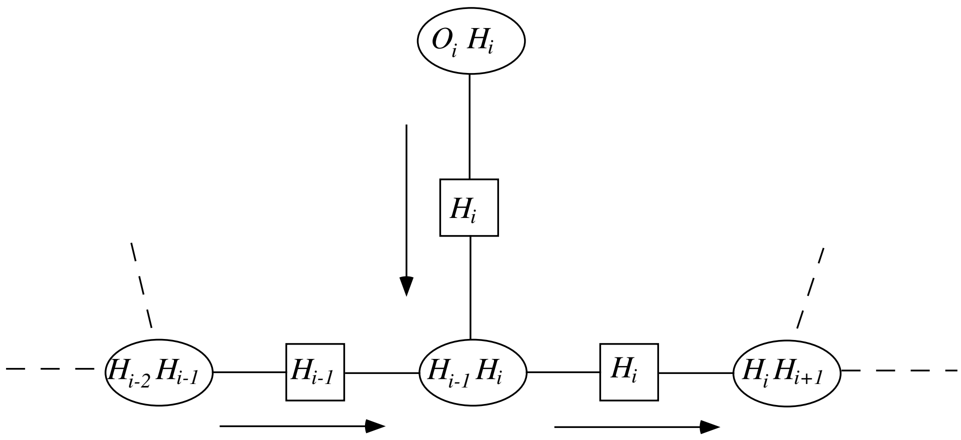

Figure 8: Local message passing in the HMM(1,1) junction tree during the collect phase of a "left to right" schedule. Ovals indicate cliques, boxes indicate separators, and arrows indicate flows.

This update factor is "absorbed" into (H_{i-1}, H_i) as follows:

$$f^*_{O_i}(h_{i-1}, h_i) = p(h_{i-1}, h_i)\lambda_{O_i}(h_i) = p(h_{i-1}, h_i)p(o^*_i|h_i) \tag{20}$$

Now consider the flow from clique (H_{i-2}, H_{i-1}) to clique (H_{i-1}, H_i). Let $\Phi_{i,j} = \{O_i, \ldots, O_j\}$ denote a set of consecutive observable variables and $\phi^*_{i,j} = \{o^*_i, \ldots, o^*_j\}$ denote a set of observed values for these variables, $1 \leq i < j \leq N$. Assume that the potential on the separator H_{i-1} has been updated to

$$f^*_{\Phi_{1,i-1}}(h_{i-1}) = p^*(h_{i-1}, \phi^*_{1,i-1}) \tag{21}$$

via earlier flows in the schedule. Thus, the update factor on separator H_{i-1} becomes

$$\lambda_{\Phi_{1,i-1}}(h_{i-1}) = \frac{p^*(h_{i-1}, \phi^*_{1,i-1})}{p(h_{i-1})} \tag{22}$$

and this gets absorbed into clique (H_{i-1}, H_i) to produce

$$\begin{aligned} f^*_{\Phi_{1,i}}(h_{i-1}, h_i) &= f^*_{O_i}(h_{i-1}, h_i)\lambda_{\Phi_{1,i-1}}(h_{i-1}) \\ &= p(h_{i-1}, h_i)p(o^*_i|h_i)\frac{p^*(h_{i-1}, \phi^*_{1,i-1})}{p(h_{i-1})} \\ &= p(o^*_i|h_i)p(h_i|h_{i-1})p^*(h_{i-1}, \phi^*_{1,i-1}). \end{aligned} \tag{23}$$

Finally, we can calculate the new potential on the separator for the flow from clique (H_{i-1}, H_i) to (H_i, H_{i+1}),

$$f^*_{\Phi_{1,i}}(h_i) = \sum_{h_{i-1}} f^*_{\Phi_{1,i}}(h_{i-1}, h_i) \tag{24}$$

$$= p(o^*_i|h_i) \sum_{h_{i-1}} p(h_i|h_{i-1})p^*(h_{i-1}, \phi^*_{1,i-1}) \tag{25}$$

$$= p(o^*_i|h_i) \sum_{h_{i-1}} p(h_i|h_{i-1})f^*_{\Phi_{1,i-1}}(h_{i-1}) \tag{26}$$

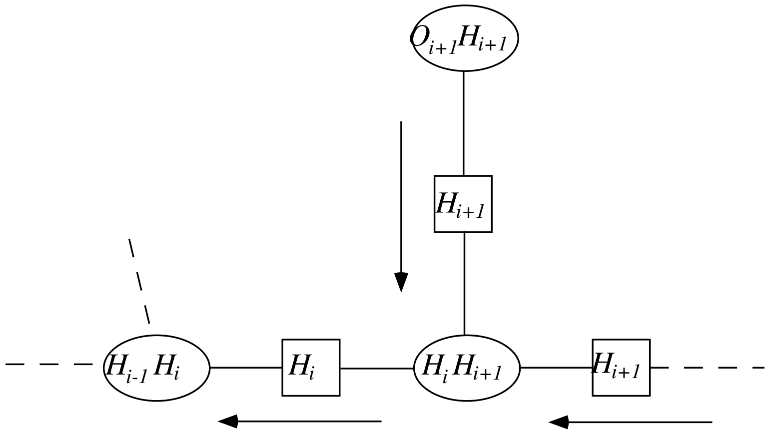

Figure 9: Local message passing in the HMM(1,1) junction tree during the collect phase of a "right to left" schedule. Ovals indicate cliques, boxes indicate separators, and arrows indicate flows.

Proceeding recursively in this manner one finally obtains at the root clique

$$f^*_{\Phi_{1,N}}(h_{N-1}, h_N) = p(h_{N-1}, h_N, \phi^*_{1,N}) \tag{27}$$

from which one can get the likelihood of the evidence,

$$p(e) = p(\phi^*_{1,N}) = \sum_{h_{N-1}, h_N} f^*_{\Phi_{1,N}}(h_{N-1}, h_N). \tag{28}$$

We note that Equation 26 directly corresponds to the recursive equation (Equation 20 in Rabiner (1989)) for the α variables used in the forward phase of the F-B algorithm, the standard HMM(1,1) inference algorithm. In particular, using a "left-to-right" schedule the updated potential functions on the separators between the hidden cliques, the $f^*_{\Phi_{1,i}}(h_i)$ functions, are exactly the α variables. Thus, when applied to HMM(1,1), the JLO algorithm produces exactly the same local recursive calculations as the forward phase of the F-B algorithm.

One can also show an equivalence between the *backward* phase of the F-B algorithm and the JLO inference algorithm. Let the "left-most" clique in the chain, (H_1, H_2), be the root clique and define a schedule such that the flows go from right to left. Figure 9 shows a local portion of the clique tree and the associated flows. Consider that the potential on clique (H_i, H_{i+1}) has been updated already by earlier flows from the right. Thus, by definition,

$$f^*_{\Phi_{i+1,N}}(h_i, h_{i+1}) = p(h_i, h_{i+1}, \phi^*_{i+1,N}). \tag{29}$$

The potential on the separator between (H_i, H_{i+1}) and (H_{i-1}, H_i) is calculated as:

$$f^*_{\Phi_{i+1,N}}(h_i) = \sum_{h_{i+1}} p(h_i, h_{i+1}, \phi^*_{i+1,N}) \tag{30}$$

$$= p(h_i) \sum_{h_{i+1}} p(h_{i+1}|h_i) p(o^*_{i+1}|h_{i+1}) p(\phi^*_{i+2,N}|h_{i+1}) \tag{31}$$

(by virtue of the various conditional independence relations in HMM(1,1))

$$= p(h_i) \sum_{h_{i+1}} p(h_{i+1}|h_i) p(o^*_{i+1}|h_{i+1}) \frac{p(\phi^*_{i+2,N}, h_{i+1})}{p(h_{i+1})} \quad (32)$$

$$= p(h_i) \sum_{h_{i+1}} p(h_i|h_{i+1}) p(o^*_{i+1}|h_{i+1}) \frac{f^*_{\Phi_{i+2,N}}(h_{i+1})}{p(h_{i+1})} \quad (33)$$

Defining the update factor on this separator yields

$$\lambda^*_{\Phi_{i+1,N}}(h_i) = \frac{f^*_{\Phi_{i+2,N}}(h_i)}{p(h_i)} \quad (34)$$

$$= \sum_{h_{i+1}} p(h_i|h_{i+1}) p(o^*_{i+1}|h_{i+1}) \frac{f^*_{\Phi_{i+2,N}}(h_{i+1})}{p(h_{i+1})} \quad (35)$$

$$= \sum_{h_{i+1}} p(h_i|h_{i+1}) p(o^*_{i+1}|h_{i+1}) \lambda^*_{\Phi_{i+2,N}}(h_{i+1}). \quad (36)$$

This set of recursive equations in λ corresponds exactly to the recursive equation (Equation 25 in Rabiner (1989)) for the β variables in the backward phase of the F-B algorithm. In fact, the update factors λ on the separators are exactly the β variables. Thus, we have shown that the JLO inference algorithm recreates the F-B algorithm for the special case of the HMM(1,1) probability model.

6.2 Equivalence of Dawid's Propagation Algorithm for Identifying MAP Assignments and the Viterbi Algorithm

Consider that one wishes to calculate $\hat{f}(\mathbf{u^h}, e) = \max_{x_1,\dots,x_K} p(x_1,\dots,x_K,e)$ and one also wishes to identify a set of values of the unobserved variables which achieve this maximum, where K is the number of unobserved (hidden) variables. This calculation can be acheived using a local propagation algorithm on the junction tree if one makes two modifications to the standard JLO inference algorithm described above. This algorithm is due to Dawid (1992) and as pointed out earlier this is the most general algorithm from a set of related methods.

Firstly, during a flow, the marginalization of the separator is replaced by:

$$\hat{b}_S(x_S) = \max_{C\setminus S} a_C(x_C) \quad (37)$$

where C is the originating clique for the flow. The definition for $\lambda_S(x_S)$ is also changed in the obvious manner.

Secondly, marginalization within a clique is replaced by maximization:

$$\hat{f}_C = \max_{\mathbf{u}\setminus x_C} p(\mathbf{u}). \quad (38)$$

Given these two changes it can be shown that if the same propagation operations are carried out as described earlier, the resulting representation $\hat{K}_f$ at equilibrium is such that the potential function on each clique C is

$$\hat{f}(x_C) = \max_{\mathbf{u}^h\setminus x_C} p(x^h_C, e, \{\mathbf{u}^h \setminus x_C\}) \quad (39)$$

where x_C^h denotes a value assignment of the hidden (unobserved) variables in clique C. Thus, once the $\hat{K}_f$ representation is obtained, one can locally identify the values of X_C^h which maximize the full joint probability as

$$\hat{x}_C^h = \arg_{x_C^h} \hat{f}(x_C). \tag{40}$$

In the probabilistic expert systems literature this procedure is known as generating the "most probable explanation" (MPE) given the observed evidence.

The HMM(1,1) MAP problem consists of being given a set of values for the observable variables, $e = \{O_1 = o_1, O_2 = o_2, \ldots, O_N = o_N\}$ and inferring

$$\max_{h_1,\ldots,h_N} p(h_1, \ldots, h_N, e). \tag{41}$$

or the set of arguments which acheive this maximum. Since Dawid's algorithm is applicable to any junction tree it can directly be applied to the HMM(1,1) junction tree in Figure 5b. In the Appendix it is shown that Dawid's algorithm, when applied to HMM(1,1), is exactly equivalent to the standard Viterbi algorithm.

6.3 Discussion of the Equivalences between the HMM and JLO Algorithms

As shown above, when HMM(1,1) is modeled as a PIN, the JLO local propagation algorithms (henceforth referred to as "the graphical algorithms") for this PIN are equivalent to the well-known F-B and Viterbi algorithms. In itself, this equivalence is not surprising since both pairs of algorithms are solving exactly the same problem via local recursive updating. For example, Dawid's method and the Viterbi algorithm are both direct applications of dynamic programming to the MAP problem.

What is interesting about this equivalence result is that the graphical algorithms are more general than the F-B and Viterbi algorithms:

1. While special purpose extensions to the standard Viterbi and F-B algorithms can be derived to handle various extensions to HMM(1,1) (Tao 1992), the JLO algorithms provide by definition a completely general exact inference method for any PIN.

2. The graphical algorithms can easily handle other inference tasks besides just calculating the likelihood of the evidence or the MAP solution. For example, missing or probabilistic evidence, simulating values from the model, calculating partial solutions, are all easy to specify in terms of the graphical algorithms. These problems in principle could also be handled by appropriate modifications to the standard F-B and Viterbi algorithms: the point is that the graphical algorithms provide the natural and direct framework for identifying such solutions.

Note that the obvious structural equivalence between PIN structures and HMMs has been noted before by Buntine (1994), Frasconi and Bengio (1994), and Lucke (1995) among others: however, the demonstration of equivalence of specific inference algorithms is new as far as we are aware.

Using the graphical algorithms on HMM(1,1), when evidence is entered into the observable states and assuming m discrete states per hidden variable, the computational complexity of solving the inference and MAP problems is $O(Nm^2)$. Naturally, given that they are equivalent for HMM(1,1), this is the same complexity as the standard F-B and Viterbi algorithms.

7 Inference and MAP Algorithms for UPINs

In Section 5 we described the JLO algorithm for local inference given a DPIN: for UPINs the procedure is very similar except for two changes to the overall algorithm. The first is the trivial observation that the moralization step is not necessary. The second difference, initialization of the junction tree is less trivial. In Section 5.2 we described how to go from a specification of conditional probabilities in a directed graph to an initial potential function representation on the cliques in the junction tree. To utilize *undirected* links in the model specification process requires new machinery to perform the initialization step. In particular we wish to compile the model into the standard form of a product of potentials on the cliques of a triangulated graph (cf. Equation 1):

$$P(\mathbf{u}) = \prod_{C \in V_C} a_C(x_C).$$

Once this initialization step has been achieved, the JLO propagation procedure proceeds as before.

Consider the chordless cycle shown in Figure 4b. Suppose that we parameterize the probability distribution on this graph by specifying pairwise marginals (or pairwise potentials) on the four pairs of neighboring nodes. We wish to convert such a local specification into a globally consistent joint probability distribution, i.e., a marginal representation. An algorithm known as *Iterative Proportional Fitting* (IPF) is available to perform this conversion. Classically, IPF proceeds as follows (Bishop, Fienberg, & Holland, 1973). Suppose for simplicity that all of the random variables are discrete (a Gaussian version of IPF is also available (Whittaker 1990)) such that the joint distribution can be represented as a table. The table is initialized with equal values in all of the cells. For each marginal in turn, the table is then rescaled by multiplying every cell by the ratio of the desired marginal to the corresponding marginal in the current table. The algorithm visits each marginal in turn, iterating over the set of marginals. If the set of marginals are consistent with a single joint distribution, the algorithm is guaranteed to converge to the joint distribution. Once the joint is available, the potentials in Equation 1 can be obtained (in principle) by marginalization.

Although IPF solves the initialization problem in principle, it is inefficient. Jiřousek and Přeučil (1995) developed an efficient version of IPF that both avoids the need for storing the joint distribution as a table and avoids the need for explicit marginalization of the joint to obtain the clique potentials. Jiřousek's version of IPF represents the evolving joint distribution directly in terms of junction tree potentials. The algorithm proceeds as follows. Let $\mathcal{I}$ be a set of subsets of V. For each $I \in \mathcal{I}$, let $q(x_I)$ denote the desired marginal on the subset I. Let the joint distribution be represented as a product over junction tree potentials (Equation 1), where each a_C is initialized to an arbitrary constant. Visit each $I \in \mathcal{I}$ in turn, updating the corresponding clique potential a_C (i.e, that potential a_C for which $I \subseteq C$) as follows:

$$a_C^*(x_C) = a_C(x_C)\frac{q(x_I)}{p(x_I)}.$$

The marginal $p(x_I)$ is obtained via the JLO algorithm, using the current set of clique potentials. Intelligent choices can be made for the order in which to visit the marginals to minimize the amount of propagation needed to compute $p(x_I)$. This algorithm is simply an efficient way of organizing the IPF calculations and inherits the latter's guarantees of convergence.

For mixed (or chain) graphs, the clique potentials are initialized to constant values and are multiplied by the appropriate conditional probability distributions associated with the directed

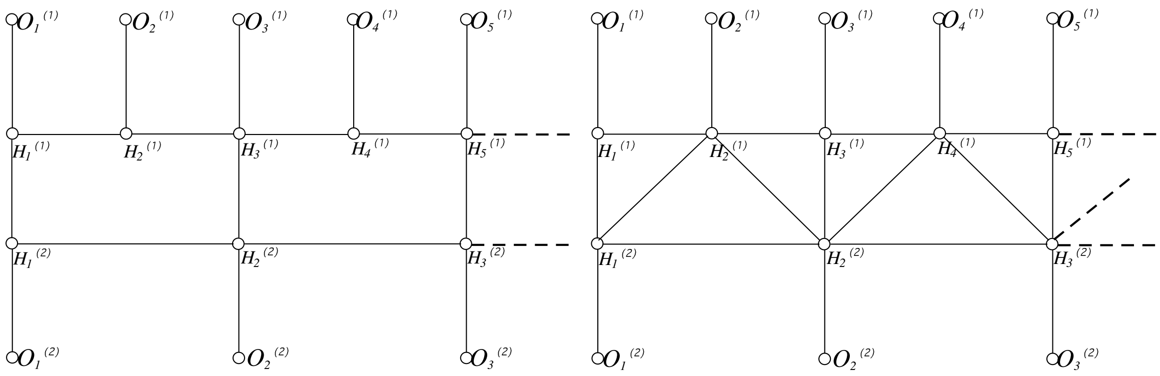

Figure 10: (a) the UPIN structure for the HMM(1,2) model with $\tau = 2$, (b) a triangulation of this UPIN structure.

links (if any). The marginals associated with the undirected links are then incorporated into the clique potentials by running IPF.

8 More Complex HMMs for Speech Modeling

Although hidden Markov models have provided an exceedingly useful framework for the modeling of speech signals, it is also true that the simple HMM(1,1) model underlying the standard framework has strong limitations as a model of speech. Real speech is generated by a set of coupled dynamical systems (lips, tongue, glottis, lungs, air columns, etc.), each of which obeys particular dynamical laws. This coupled physical process is not well modeled by the unstructured state transition matrix of HMM(1,1). Moreover, the first-order Markov properties of HMM(1,1) are not well suited to modeling the ubiquitous coarticulation effects that occur in speech, particularly coarticulatory effects that extend across several phonemes (cf. Kent & Minifie, 1977). A variety of techniques have been developed to surmount these basic weaknesses of the HMM(1,1) model, including mixture modeling of emission probabilities, triphone modeling, and discriminative training. All of these methods, however, leave intact the basic probabilistic structure of HMM(1,1) as expressed by its PIN structure.

In this section we describe several extensions of HMM(1,1) that assume additional probabilistic structure beyond that assumed by HMM(1,1). PINs provide a key tool in the study of these more complex models. The role of PINs is twofold: first, they provide a concise description of the probabilistic dependencies assumed by a particular model, and second, they provide a general algorithm for computing likelihoods. This second property is particularly important—the existence of the JLO algorithm frees us from having to derive particular recursive algorithms on a case-by-case basis.

The first model that we consider can be viewed as a coupling of two HMM(1,1) chains (Saul & Jordan, 1995). Such a model can be useful in general sensor fusion problems, for example in the fusion of an audio signal with a video signal in lipreading. Because different sensory signals generally have different bandwidths, it may be useful to couple separate Markov models that are developed specifically for each of the individual signals. The alternative is to force the problem into an HMM(1,1) framework by either oversampling the slower signal, which requires additional parameters and leads to a high-variance estimator, or downsampling the faster signal, which generally oversmoothes the data and yields a biased estimator. Consider the HMM(1,2) structure shown in Figure 10a. This model involves two HMM(1,1) backbones that are coupled together via undirected links between the state variables. Let $H_i^{(1)}$ and $O_i^{(1)}$ denote the i^{th} state and i^{th} output of the "fast" chain, respectively, and let $H_i^{(2)}$ and $O_i^{(2)}$ denote the i^{th} state and i^{th} output of the "slow" chain. Suppose that the fast chain is sampled τ times as often as the slow chain. Then $H_{i'}^{(1)}$ is connected to $H_i^{(2)}$ for i' equal to $\tau(i-1)+1$. Given this value for i', the Markov model for the coupled chain implies the following conditional independencies for the state variables:

$$\{H_{i'}^{(1)}, H_i^{(2)}\} \perp \{H_1^{(1)}, O_1^{(1)}, H_1^{(2)}, O_1^{(2)}, \ldots, H_{i'-2}^{(1)}, O_{i'-2}^{(1)}, H_{i-2}^{(2)}, O_{i-2}^{(2)}, O_{i'-1}^{(1)}, O_{i-1}^{(2)}\} | \{H_{i'-1}^{(1)}, H_{i-1}^{(2)}\}, \tag{42}$$

as well as the following conditional independencies for the output variables:

$$\{O_{i'}^{(1)}, O_i^{(2)}\} \perp \{H_1^{(1)}, O_1^{(1)}, H_1^{(2)}, O_1^{(2)}, \ldots, H_{i'-1}^{(1)}, O_{i'-1}^{(1)}, H_{i-1}^{(2)}, O_{i-1}^{(2)}\} | \{H_{i'}^{(1)}, H_i^{(2)}\}. \tag{43}$$

Additional conditional independencies can be read off the UPIN structure (see Figure 10a).

As is readily seen in Figure 10a, the HMM(1,2) graph is not triangulated, thus the HMM(1,2) probability model is not decomposable. However, the graph can be readily triangulated to form a decomposable cover for the HMM(1,2) probability model (see Section 3.1.2). The JLO algorithm provides an efficient algorithm for calculating likelihoods in this graph. This can be seen in Figure 10b, where we show a triangulation of the HMM(1,2) graph. The triangulation adds $O(N_h)$ links to the graph (where N_h is the number of hidden nodes in the graph) and creates a junction tree in which each clique is a cluster of three state variables from the underlying UPIN structure. Assuming m values for each state variable in each chain, we obtain an algorithm whose time complexity is $O(N_h m^3)$. This can be compared to the naive approach of transforming the HMM(1,2) model to a Cartesian product HMM(1,1) model, which not only has the disadvantage of requiring subsampling or oversampling, but also has a time complexity of $O(N_h m^4)$.

Directed graph semantics can also play an important role in constructing interesting variations on the hidden Markov model theme. Consider Figure 11a, which shows an HMM(1,2) model in which a single output stream is coupled to a pair of underlying state sequences. In a speech modeling application such a structure might be used to capture the fact that a given acoustic pattern can have multiple underlying articulatory causes. For example, equivalent shifts in formant frequencies can be caused by lip-rounding or tongue-raising; such phenomena are generically refered to as "trading relations" in the speech psychophysics literature (Lindblom 1990; Perkell et al. 1993). Once a particular acoustic pattern is observed, the causes become dependent; thus for example, evidence that the lips are rounded would act to discount inferences that the tongue has been raised. These inferences propagate forward and backward in time and couple the chains. Formally, these induced dependencies are accounted for by the links added between the state sequences during the moralization of the graph (see Figure 11b). This figure shows that the underlying calculations for this model are closely related to those of the earlier HMM(1,2), but the model specification is very different in the two cases.

Saul and Jordan (1996) have proposed a second extension of the HMM(1,1) model which is motivated by the desire to provide a more effective model of coarticulation (see also Stolorz,

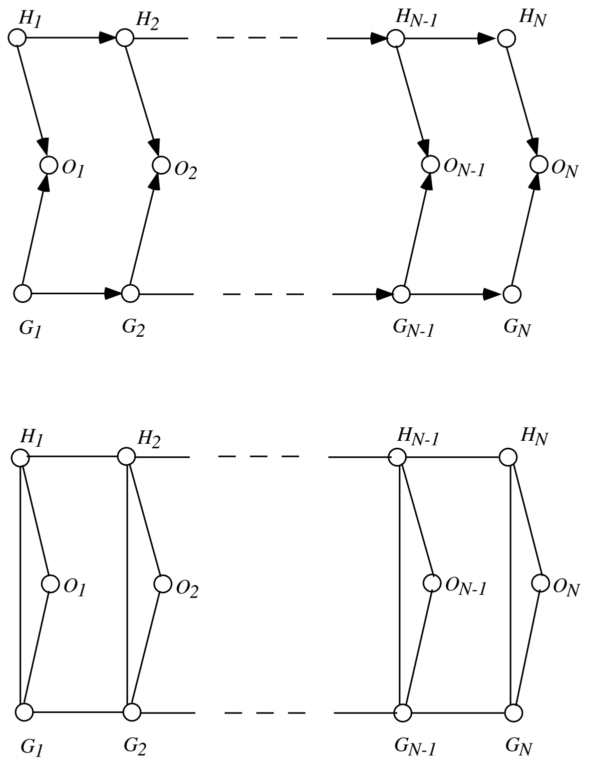

Figure 11: (a) the DPIN structure for HMM(1,2) with a single observable sequence coupled to a pair of underlying state sequences, (b) the moralization of this DPIN structure.

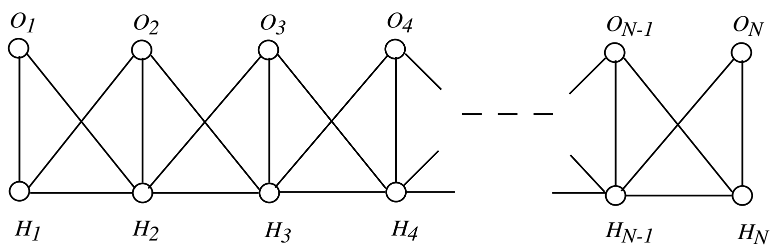

Figure 12: The UPIN structure for HMM(3,1).

1994). In this model, shown in Figure 12, coarticulatory influences are modeled via additional links between output variables and states along an HMM(1,1) backbone. One approach to performing calculations in this model is to treat it as a K^{th}-order Markov chain, and transform it into an HMM(1,1) model by defining higher-order state variables. A graphical modeling approach is more flexible—it is possible for example to introduce links between states and outputs K time steps apart without introducing links for the intervening time intervals. More generally, the graphical modeling approach to the HMM(K,1) model allows the specification of different interaction matrices at different time scales; this is awkward in the K^{th}-order Markov chain formalism.

The HMM(3,1) graph is triangulated as is, and thus, the time complexity of the JLO algorithm is therefore $O(N_h m^3)$. In general a HMM(K,1) graph creates cliques of size $O(m^K)$ and the JLO algorithm runs in time $O(N_h m^K)$.

As these examples suggest, the graphical modeling framework provides a useful framework for exploring extensions of hidden Markov models. The examples also make clear, however, that the graphical algorithms are no panacea. The m^K complexity of HMM(K,1) will be prohibitive for large K. Also, the generalization of HMM(1,2) to HMM(1,K) (couplings of K chains) is intractable. Recent research has therefore focused on approximate algorithms for inference in such structures—see Saul and Jordan (1996) for HMM(K,1) and Ghahramani and Jordan (1996) and Williams and Hinton (1990) for HMM(1,K). These authors have developed an approximation methodology based on mean-field theory from statistical physics. While discussion of mean-field algorithms is beyond the scope of this paper, it is worth noting that the graphical modeling framework plays a useful role in the development of these approximations. Essentially the mean-field approach involves creating a simplified graph for which tractable algorithms are available, and minimizing a probabilistic distance between the tractable graph and the intractable graph. The JLO algorithm is called as a subroutine on the tractable graph during the minimization process.

9 Learning and PINs

9.1 Parameter Estimation for PINs

The parameters of a graphical model can be estimated with maximum-likelihood (ML), maximum-a-posteriori (MAP), or full Bayesian methods, using traditional techniques such as gradient descent, expectation-maximization (EM) (e.g., Dempster et al., 1977), and Monte-Carlo sampling (e.g., Neal, 1993). For the standard HMM(1,1) model discussed in this paper, where either discrete, Gaussian, or Gaussian-mixture codebooks are used, a ML or MAP estimate using EM is a well-known efficient approach (Poritz 1988; Rabiner 1989). An important aspect of the application of the EM algorithm to PINs is that the JLO algorithm can be used to perform the E step.

For purposes of illustration, and in keeping with the rest of the paper, let us consider the case where all variables in $\mathbf{U}$ are discrete. Let x^k and $pa(X)^j$ denote the kth state of variable X and jth state of variables $pa(X)$, respectively. Suppose we have a directed HMM-like model M (a DPIN) with mutually independent parameters $\boldsymbol{\theta} = \cup_{jk}\{\theta_{Hjk}, \theta_{Ojk}\}$, where $\theta_{Hjk} = p(h_i^k|pa(H_i)^j, M)$ and $\theta_{Ojk} = p(o_i^k|pa(O_i)^j, M)$ for all i. In addition, suppose we have observed data $D = \{e_1, \ldots, e_S\}$, an (iid) random sample from the true distribution.

The EM algorithm finds a local maximum of the likelihood $p(D|\boldsymbol{\theta}, M)$ by initializing the parameters $\boldsymbol{\theta}$ (e.g., at random or via some clustering algorithm) and repeating E and M steps.

In the E step, we compute the expected sufficient statistic for each of the parameters, given D and the current values for $\boldsymbol{\theta}$. Let S_{Hjk} be the sufficient statistic for θ_{Hjk}. The expected sufficient statistic $E(S_{Hjk}|D, \boldsymbol{\theta}, M)$ is given by

$$E(S_{Hjk}|D, \boldsymbol{\theta}, M) = \sum_{l=1}^{S} \sum_{i} p(h_i^k, pa(H_i)^j | e_l, \boldsymbol{\theta}, M)$$

As mentioned, an important feature of the EM algorithm applied to PINs is that each term in the sum can be computed using the JLO algorithm. The expected sufficient statistic for θ_{Ojk} is computed similarly. In the M step, we use the expected sufficient statistics as if they were actual sufficient statistics, and set the new values of $\boldsymbol{\theta}$ to be those that maximize the likelihood of these statistics:

$$\theta_{Hjk} = \frac{E(S_{Hjk}|D, \boldsymbol{\theta}, M)}{\sum_k E(S_{Hjk}|D, \boldsymbol{\theta}, M)} \qquad \theta_{Ojk} = \frac{E(S_{Ojk}|D, \boldsymbol{\theta}, M)}{\sum_k E(S_{Ojk}|D, \boldsymbol{\theta}, M)}$$

The EM algorithm also can be used to find a local maximum of the posterior probability $p(\boldsymbol{\theta}|D, M) \propto p(D|\boldsymbol{\theta}, M) \cdot p(\boldsymbol{\theta}|M)$, where $p(\boldsymbol{\theta}|M)$ is the parameter prior. Priors most often used are conjugate distributions, such as the Dirichlet distribution for the parameters of discrete variables and the mixing coefficients of Gaussian-mixture codebooks, and the normal-Wishart distribution for the parameters of Gaussian codebooks (DeGroot 1970; Buntine 1994; Heckerman and Geiger 1995). These priors have also been used in MAP estimates of standard HMMs (e.g., Gauvain and Lee, 1994). Heckerman and Geiger (1995) describe a simple method for assessing these priors.

The use of the EM algorithm for UPINs is similar. Suppose that the undirected model M consists of cliques C_{ij} such that the parameters of $C_{i_1 j}$ and $C_{i_2 j}$ are the same for any i_1 and i_2. That is, suppose $p(C_{i,j} = c_{i,j}^k | M) = \theta_{jk}$ for all i. In addition, suppose that the parameters $\boldsymbol{\theta} = \cup_{jk} \theta_{jk}$ are mutually independent. In this case, we can estimate the parameters for the clique marginals, and then use Jiřousek's IPF algorithm on a triangulation of M to compute a consistent estimate of the joint distribution. As in the directed case, we can use the JLO algorithm to perform the E step:

$$E(S_{jk}|D, \boldsymbol{\theta}, M) = \sum_{l=1}^{S} \sum_{i} p(c_{i,j}^k | e_l, \boldsymbol{\theta}, M)$$

9.2 Model Selection and Averaging for PINs

In some situations it is useful to use data to guide the selection of an appropriate model. For example, the presence of some arcs or the number of states of a hidden variable may be in doubt. One solution to this problem is the Bayesian approach, in which we assign prior probabilities $p(M)$ to different models, and compute their relative posterior probabilities given data:

$$p(M|D) \propto p(M)\ p(D|M) = p(M) \int p(D|\boldsymbol{\theta}, M)\ p(\boldsymbol{\theta}|M)\ d\boldsymbol{\theta} \tag{44}$$

We then select the model with the highest posterior probability, or average the predictions of two or more models weighted by their relative posterior probabilities.

When data is missing—for example, when some variables are hidden—the exact computation of the integral in Equation 44 is usually intractable. Nonetheless, simple approximations to this integral exist, such as the Bayesian Information Criterion (BIC) described by Schwarz (1978):

$$\log p(D|M) \approx \log p(D|\hat{\boldsymbol{\theta}}, M) - \frac{d}{2}\log S$$

where $\hat{\boldsymbol{\theta}}$ is the ML estimate, S is the number of cases in D, and d is the dimension of M—typically, the number of parameters of M. The first term of this "score" for M rewards how well the data fits M, whereas the second term punishes model complexity. Note that this score does not depend on the parameter prior, and thus can be applied easily.[1] For examples of applications of BIC in the context of PINs and other statistical models, see Raftery (1995).

The BIC score is the additive inverse of Rissanen's (1987) minimum description length (MDL). Other scores, which can be viewed as approximations to the marginal likelihood, are hypothesis testing (Raftery 1995) and cross validation (Fung and Crawford 1990). Buntine (in press) provides a comprehensive review of the literature on learning PINs.

In the context of HMM(K, J) type structures, an obvious question is how one could learn such structure from data, where K and J are unknown a priori. From a model identification viewpoint, this is an easier problem than that of learning an arbitrary PIN, because the possible models under consideration are highly constrained. Thus, using both the estimation techniques for a particular model described in the previous section (and the JLO algorithm for solving the E-step as described in detail earlier in the paper), and the Bayesian (and alternative) model selection procedures outlined above, the algorithmic prescriptions for learning such models in a principled fashion are already in place.

10 Summary

Probabilistic independence networks provide a useful framework for both the analysis and application of multivariate probability models when there is considerable structure in the model in the form of conditional independence. The graphical modelling approach both clarifies the independence semantics of the model and yields efficient computational algorithms for probabilistic inference. This paper has shown that it is useful to cast HMM structures in a graphical model framework. In particular, the well known F-B and Viterbi algorithms were shown to be special cases of more general algorithms from the graphical modelling literature. Furthermore, more complex HMM structures, beyond the traditional first-order model, can be analyzed profitably and directly using generally-applicable graphical modeling techniques.

References

Bishop, Y.M.M., Fienberg, S.E. and Holland, P.W. 1973. *Discrete Multivariate Analysis: Theory and Practice.* MIT Press, Cambridge MA.

Buntine, W. 1994. Operations for learning with graphical models. *Journal of Artificial Intelligence Research.* **2** 159–225.

Buntine, W. in press. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering.*

[1] One caveat: The BIC score is derived under the assumption that the parameter prior is positive throughout its domain.

Dawid, A. P. 1992. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing.* **2** 25–36.

DeGroot, M. 1970. *Optimal Statistical Decisions.* McGraw-Hill, New York.

Dempster, A., Laird, N., Rubin, D.1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B.* **39**, 1–38.

Frasconi, P. and Bengio, Y. 1994. An EM approach to grammatical inference: input/output HMMs. *Proceedings of the 12th IAPR Intl. Conf. on Pattern Recognition*, IEEE Computer Society Press. 289–294.

Fung, R. M. and Crawford, S. L. 1990. A system for induction of probabilistic models. *Eighth National Conference on Artificial Intelligence*, Boston, MA: AAAI, 762-779.

Gauvain, J., Lee, C. 1994. Maximum *a posteriori* estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Trans. Sig. Audio Proc.*. **2**, 291–298.

Geman, S. and Geman, D. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Patt. Anal. Mach. Intell.* **6**, 721-741.

Ghahramani, Z., and Jordan, M. I. 1996. Factorial Hidden Markov models. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8*, MIT Press, Cambridge MA.

Heckerman, D., and Geiger, D. 1995. Likelihoods and priors for Bayesian networks. MSR-TR-95-54, Microsoft, Redmond, WA.

Heckerman, D., Geiger, D., and Chickering, D. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning.* **20**, 197–243.

Isham, V. 1981. An introduction to spatial point processes and Markov random fields. *International Statistical Review.* **49**, 21-43.

Jensen, F. V. 1995. *Introduction to Bayesian networks.* HUGIN Expert A/S, Aalborg, Denmark.

Jensen, F. V., Lauritzen, S. L. and Olesen, K. G., 1990. Bayesian updating in recursive graphical models by local computations. *Computational Statistical Quarterly.* **4**, 269–282.

Jiřousek, R. and Přeučil, S. 1995. On the effective implementation of the iterative proportional fitting procedure. *Computational Statistics and Data Analysis.* **19**, 177–189.

Kent, R. D. & Minifie, F. D. 1977. Coarticulation in recent speech production models. *Journal of Phonetics.* **5**, 115-117.

Lauritzen, S. L. and Spiegelhalter D. J. 1988. Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *J. Roy. Statist. Soc. Ser. B.* **50** 157–224.

Lauritzen, S., and Wermuth, N. 1989. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics.* **17**, 31–57.

Lauritzen, S. L., Dawid, A. P., Larsen, B. N., and Leimer, H. G 1990. Independence properties of directed Markov fields. *Networks.* **20**, 491–505.

Lindblom, B. 1990. Explaining phonetic variation: A sketch of the H&H theory. In *Speech Production and Speech Modeling*, W.J. Hardcastle and A. Marchal, (Eds.). Kluwer: Dordrecht.

Lucke, H. 1995. Bayesian Belief Networks as a tool for stochastic parsing. *Speech Communication.* **16**, 89–118.

Modestino, J. and Zhang, J. 1992. A Markov random field model-based approach to image segmentation. *IEEE Trans. Patt. Anal. Mach. Int.* **14**(6), 606–615.

Neal, R. 1993. Probabilistic inference using Markov chain Monte Carlo methods. CRG-TR-93-1, Department of Computer Science, University of Toronto.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, San Mateo, CA: Morgan Kaufmann Publishers.

Pearl, J., Geiger, D., and Verma, T. 1990. The logic of influence diagrams. *Influence Diagrams, Belief Nets, and Decision Analysis.* Oliver, R. M. and Smith, J. Q. (eds.). Chichester, U.K.: John Wiley and Sons. 67–83.

Perkell, J. S., Matthies, M. L., Svirsky, M. A., and Jordan, M. I. 1993. Trading relations between tongue-body raising and lip rounding in production of the vowel /u/: A pilot motor equivalence study. *Journal of the Acoustical Society of America.* **93**, 2948-2961.

Poritz, A. M. 1988. Hidden Markov models: a guided tour. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing.* New York: IEEE Press. vol.1, 7-13.

Rabiner, L., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**, 257-285.

Raftery, A. 1995. Bayesian model selection in social research. In Marsden, P., *Sociological Methodology.* Blackwells, Cambridge, MA.

Rissanen, J. 1987. Stochastic complexity (with discussion). *Journal of the Royal Statistical Society, Series B.* **49**, 223–239 and 253–265.

Saul, L. K., and Jordan, M. I. 1995. Boltzmann chains and hidden Markov models. In G. Tesauro, D. S. Touretzky & T. K. Leen, (Eds.), *Advances in Neural Information Processing Systems 7*, MIT Press, Cambridge MA.

Saul, L. K., and Jordan, M. I. 1996. Exploiting tractable substructures in intractable networks. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8*, MIT Press, Cambridge MA.

Shachter, R. D., Anderson, S. K. and Szolovits, P. 1994. Global conditioning for prababilistic inference in belief networks. *Proceedings of the Uncertainty in AI Conference 1994*, San Francisco, CA: Morgan Kaufmann, 514–522.

Schwarz, G. 1978. Estimating the dimension of a model. *Annals of Statistics.* **6**, 461–464.

Smyth, P. 1994. Hidden Markov models for fault detection in dynamic systems. *Pattern Recognition*, **27**, 149-164.

Spiegelhalter, D. J., Dawid, A. P., Hutchinson, T. A., and Cowell, R. G. 1991. Probabilistic expert systems and graphical modelling: a case study in drug safety. *Phil. Trans. R. Soc. Lond. A.* 337, 387–405.

Spirtes, P. and Meek, C. 1995. Learning Bayesian networks with discrete variables from data. In *Proceedings of First International Conference on Knowledge Discovery and Data Mining,* Montreal, QU. Morgan Kaufmann.

Stolorz, P. 1994. Recursive approaches to the statistical physics of lattice proteins. In L. Hunter, ed. *Proc. 27th Hawaii Intl. Conf. on System Sciences*, **V**, 316-325.

Tao, C., 1992. A generalization of the discrete hidden Markov model and of the Viterbi algorithm. *Pattern Recognition*, **25**(11), 1381–1387.

Vandermeulen, D., Verbeeck, R., Berben, L. Delaere, D., Suetens, P. and Marchal, G. 1994. Continuous voxel classification by stochastic relaxation: theory and application to MR imaging and MR angiography. *Image and Vision Computing.* **12**(9) 559–572.

Whittaker, J. 1990. *Graphical Models in Applied Multivariate Statistics*, Chichester, UK: John Wiley and Sons.

Williams, C., and Hinton, G. E. 1990. Mean field networks that learn to discriminate temporally distorted strings. *Proc. Connectionist Models Summer School*, San Mateo, CA: Morgan Kaufmann, 18–22.

Appendix 1: The Viterbi Algorithm for HMM(1,1) is a Special Case of Dawid's Algorithm

As with the inference problem, let the final clique in the chain containing (H_{N-1}, H_N) be the root clique and use the same schedule, i.e., first a "left-to-right" collection phase into the root clique, followed by a "right-to-left" distribution phase out from the root clique. Again it is assumed that the junction tree has been initialized so that the potential functions are the local marginals, and the observable evidence e has been entered into the cliques in the same manner as described for the inference algorithm.

We refer again to Figure 8: the sequence of flow and absorption operations is identical to that of the inference algorithm with the exception that marginalization operations are replaced by maximization. Thus, the potential on the separator between (O_i, H_i) and (H_{i-1}, H_i) is initially updated to

$$\hat{f}_{O_i}(h_i) = \max_{o_i} p(h_i, o_i) = p(h_i, o_i^*). \tag{45}$$

The update factor for this separator is

$$\lambda_{O_i}(h_i) = \frac{p(h_i, o_i^*)}{p(h_i)} = p(o_i^* | h_i), \tag{46}$$

and after absorption into the clique (H_{i-1}, H_i) one gets

$$\hat{f}_{O_i}(h_{i-1}, h_i) = p(h_{i-1}, h_i) p(o_i^* | h_i). \tag{47}$$

Now consider the flow from clique (H_{i-2}, H_{i-1}) to (H_{i-1}, H_i). Let $H_{i,j} = \{H_i, \ldots, H_j\}$ denote a set of consecutive observable variables and $h^*_{i,j} = \{h^*_i, \ldots, h^*_j\}$, denote the observed values for these variables, $1 \le i < j \le N$. Assume that the potential on separator H_{i-1} has been updated to

$$\hat{f}_{\Phi_{1,i-1}}(h_{i-1}) = \max_{h_{1,i-2}} p(h_{i-1}, h_{1,i-2}, \phi^*_{1,i-1}) \tag{48}$$

via earlier flows in the schedule. Thus, the update factor for separator H_{i-1} becomes

$$\lambda_{\Phi_{1,i-1}}(h_{i-1}) = \frac{\max_{h_{1,i-2}} p(h_{i-1}, h_{1,i-2}, \phi^*_{1,i-1})}{p(h_{i-1})} \tag{49}$$

and this gets absorbed into clique (H_{i-1}, H_i) to produce

$$\begin{aligned} \hat{f}_{\Phi_{1,i}}(h_{i-1}, h_i) &= \hat{f}_{O_i}(h_{i-1}, h_i)\lambda_{\Phi_{1,i-1}}(h_{i-1}) && (50) \\ &= p(h_{i-1}, h_i)p(o^*_i|h_i)\frac{\max_{h_{1,i-2}} p(h_{i-1}, h_{1,i-2}, \phi^*_{1,i-1})}{p(h_{i-1})}. && (51) \end{aligned}$$

We can now obtain the new potential on the separator for the flow from clique (H_{i-1}, H_i) to (H_i, H_{i+1}),

$$\begin{aligned} \hat{f}_{\Phi_{1,i}}(h_i) &= \max_{h_{i-1}} \hat{f}_{\Phi_{1,i}}(h_{i-1}, h_i) && (52) \\ &= p(o^*_i|h_i) \max_{h_{i-1}}\{p(h_i|h_{i-1}) \max_{h_{1,i-2}} p(h_{i-1}, h_{1,i-2}, \phi^*_{1,i-1})\} && (53) \\ &= p(o^*_i|h_i) \max_{h_{1,i-1}}\{p(h_i|h_{i-1})p(h_{i-1}, h_{1,i-2}, \phi^*_{1,i-1})\} && (54) \\ &= \max_{h_{1,i-1}} p(h_i, h_{1,i-1}, \phi^*_{1,i}) && (55) \end{aligned}$$

which is the result one expects for the updated potential at this clique. Thus, we can express the separator potential $\hat{f}_{\Phi_{1,i}}(h_i)$ recursively (via Equation 54) as

$$\hat{f}_{\Phi_{1,i}}(h_i) = p(o^*_i|h_i) \max_{h_{i-1}}\{p(h_i|h_{i-1})\hat{f}_{\Phi_{1,i-1}}(h_{i-1})\}. \tag{56}$$

This is the same recursive equation as used in the δ variables in the Viterbi algorithm (Equation 33a in Rabiner (1990)): the separator potentials in Dawid's algorithm using a left-to-right schedule are exactly the same as the $\delta's$ used in the Viterbi method for solving the MAP problem in HMM(1,1).

Proceeding recursively in this manner one finally obtains at the root clique

$$\hat{f}_{\Phi_{1,N}}(h_{N-1}, h_N) = \max_{h_{1,N-2}} p(h_{N-1}, h_N, h_{N-2}, \phi^*_{1,N}) \tag{57}$$

from which one can get the likelihood of the evidence given the most likely state of the hidden variables:

$$\begin{aligned} \hat{f}(e) &= \max_{h_{N-1},h_N} \hat{f}_{\Phi_{1,N}}(h_{N-1}, h_N) && (58) \\ &= \max_{h_{1,N}} p(h_{1,N}, \phi^*_{1,N}) && (59) \end{aligned}$$

Identification of the values of the hidden variables which maximize the evidence likelihood can be carried out in the standard manner as in the Viterbi method, namely by keeping a pointer at each clique along the flow in the forward direction back to the previous clique and then backtracking along this list of pointers from the root clique after the collection phase is complete. An alternative approach is to use the distribute phase of the Dawid algorithm: this has the same effect, namely, once the distribution flows are completed, each local clique can calculate both the maximum value of the evidence likelihood given the hidden variables and the values of the hidden variables in this maximum which are local to that particular clique.