

# Structural Learning of Dynamic Bayesian Networks in Speech Recognition

TECHNICAL REPORT

Murat Deviren  
Speech Group, LORIA

13 September 2001

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Bayesian networks</b>	<b>3</b>
2.1	Definition . . . . .	3
2.2	Parameterization . . . . .	4
2.2.1	Discrete variable with discrete parents . . . . .	4
2.2.2	Continuous variable with continuous parents . . . . .	5
2.2.3	Continuous variable with discrete and continuous parents . . . . .	5
<b>3</b>	<b>Inference Engine</b>	<b>5</b>
3.1	Constructing the Junction Tree . . . . .	6
3.2	Message passing . . . . .	7
3.3	Initializing the Junction Tree . . . . .	8
3.4	Generalized operations . . . . .	9
3.4.1	Most probable configuration . . . . .	9
3.4.2	Random configuration . . . . .	10
<b>4</b>	<b>Learning parameters from data</b>	<b>10</b>
4.1	Maximum Likelihood (ML) Estimation . . . . .	10
4.1.1	Data likelihood . . . . .	10
4.1.2	ML parameter estimation for discrete BNs . . . . .	11
4.1.3	ML parameter estimation for continuous BNs . . . . .	11
4.1.4	ML parameter estimation for hybrid BNs . . . . .	12
4.2	Maximum a posteriori (MAP) estimation and Bayesian learning . . . . .	12
4.3	EM algorithm for parameter learning in the case of incomplete data . . . . .	13
4.3.1	EM algorithm for parameter learning in discrete BNs . . . . .	13
4.3.2	EM algorithm for parameter learning in continuous BNs . . . . .	14
4.3.3	EM algorithm for parameter learning in hybrid BNs . . . . .	14
<b>5</b>	<b>Learning structure from data</b>	<b>15</b>
5.1	Structural learning with complete data . . . . .	15
5.2	Scoring structure . . . . .	15
5.2.1	Bayesian Information Criteria (BIC), Minimum Description Length (MDL) Score . . . . .	15
5.2.2	Bayesian Dirichlet (BD, BDe) Score . . . . .	16
5.2.3	Structure Priors . . . . .	17
5.3	Searching structure space . . . . .	18
5.4	Structural EM algorithm with incomplete data . . . . .	18
<b>6</b>	<b>Dynamic Bayesian networks</b>	<b>19</b>
6.1	Definition . . . . .	19
6.2	Dependency range and structural inductions . . . . .	20
6.3	Inference and Learning in DBNs . . . . .	21
<b>7</b>	<b>DBNs for speech recognition</b>	<b>22</b>
7.1	DBN structures for speech recognition . . . . .	23
7.2	Learning DBNs for speech recognition . . . . .	24
7.3	Topology representation in DBNs . . . . .	26

<b>8 Experiments</b>	<b>26</b>
8.1 Isolated digit recognition . . . . .	26
<b>9 Discussion and Future Work</b>	<b>28</b>

# 1 Introduction

The use of statistical models in speech recognition has been shown to be a powerful approach. The most commonly used model is the Hidden Markov Model (HMM). Several inference and learning algorithms have been developed and several extensions have been considered for the specific task of speech recognition. HMMs are flexible such that they can be tuned by adjusting several parameters (i.e. the number of hidden states, the functional form of the output probability density, the order of the hidden process). However, the conditional independence (CI) assumptions are fixed. The flexibility of the model is limited with these assumptions. Moreover, these assumptions may not be consistent with the data. In this work we are proposing a solution to the following problem:

What are the CI assumptions that are consistent with data for the speech recognition task?

In other words, we propose a methodology to learn the CI from data. And we use Bayesian Networks for this approach. Actually HMMs are a subset of BNs. Therefore, tackling the problem in the more general setting we have more degrees of freedom. Although this freedom comes with the cost of complexity, we propose a controlled mechanism over complexity and modelling fidelity.

In Section 2, a general definition of Bayesian Networks is presented with a set of possible parameterizations of the model. Next the inference engine for BNs is explained in detail. In Section 4 and 5, we describe the methods of learning parameters and structure of BNs from data. Section 6 is devoted to Dynamic Bayesian Networks (DBN). We present the representation of DBNs in terms of static BNs according to the temporal properties of the underlying process. In Section 7, we describe the set of structures that we propose for the speech recognition task and give the details of the learning algorithm in this setting. Finally in Section 8, the experimental results are given. And in Section 9, a brief discussion of the approach is given with some future work plans.

## 2 Bayesian networks

### 2.1 Definition

A Bayesian Network (BN) is an efficient representation for the joint probability distribution (jpd) of a set of random variables based on a set of conditional independence (CI) assumptions. A BN is constructed over a set of nodes, where each node is associated with a random variable  $X_i$ . The conditional independencies among the variables are represented with a *directed acyclic graph (DAG)*. The arcs in the DAG represent dependencies. Being directional and acyclic the structure of a BN provides a direct factorization of the joint probability

distribution <sup>1</sup>.

$$P(\mathbf{X}) = \prod_{i=1}^n P(X_i|\Pi_i) \tag{2}$$

This factorization enables the use of simple local conditional distributions rather than using the full jpd for inferring probabilities. More formally, a BN can be defined as follows:

**Definition 1 (Bayesian Network)** *A Bayesian Network is a pair  $(S, \Theta)$ , where  $S$  is a DAG,  $\Theta$  is a parameterization of a set  $\{P(X_1|\Pi_1), \dots, P(X_n|\Pi_n)\}$  of conditional probability distributions, one for each variable, and  $\Pi_i$  is the set of parents of node  $X_i$  in  $S$ . The set of CPDs defines the associated jpd as,*

$$P(\mathbf{X}) = \prod_{i=1}^n P(X_i|\Pi_i). \tag{3}$$

Throughout this paper,  $X_i$  denotes a continuous or discrete random variable. Values of the random variable will be indicated by lower case letters as in  $x_i$ . For a discrete variable that takes  $r$  values,  $x_i^k$  denote a specific assignment for  $1 \leq k \leq r$ . A set of variables is denoted in boldface letters  $\mathbf{X} = \{X_1, \dots, X_n\}$ .

## 2.2 Parameterization

The selection of local conditional probability distributions (lcpd) depends on the specific problem. In this work we consider multinomial distributions for discrete variables and conditional Gaussian densities for continuous variables. In the following we give respective parameterizations of lcpds, for each possible setting of a variable and its parents. We exclude the case where a discrete variable has a continuous parent. There is no exact inference algorithm for such a case.

### 2.2.1 Discrete variable with discrete parents

For a discrete variable  $X_i$  with discrete parents  $\Pi_i$ , the lcpd is represented with a conditional probability table,  $\Theta_i$ . For each distinct configuration  $\pi_i^j$  of  $\Pi_i$ , a probability  $\theta_{ijk}$  is associated to each instance  $x_i^k$  of  $X_i$ . Hence the parameters of the lcpd are defined as follows :

$$\theta_{ijk} = P(X_i = x_i^k | \Pi_i = \pi_i^j) \quad \begin{matrix} i = 1, \dots, n \\ j = 1, \dots, q_i \\ k = 1, \dots, r_i. \end{matrix} \tag{4}$$

$r_i$  is the number of possible values that  $X_i$  can attain and  $q_i$  is the number of distinct configurations of  $\Pi_i$ .

---

<sup>1</sup>From the chain rule of probability, we have the following general factorization of the jpd of  $\mathbf{X}$  :

$$P(\mathbf{X}) = \prod_{i=1}^n P(X_i | X_{i-1}, \dots, X_1) \tag{1}$$

If  $\Pi_i$  is chosen as the set of variables that renders  $X_i$  and  $X_{i-1}, \dots, X_1$ , then we have the jpd factorization of BNs.

### 2.2.2 Continuous variable with continuous parents

For a continuous variable  $X_i$ , with continuous parents  $\Pi_i$ , the lcpd is a conditional Gaussian where the mean is a linear function of the values of the parents.

$$P(X_i|\Pi_i) = \mathcal{N}(\beta_i^T \Pi_i, \sigma_i^2) \quad (5)$$

$\beta_i$  is a measure of dependency of  $X_i$  on its parents. The size of the vector  $\beta_i$  is equal to the number of parents of  $X_i$ . Hence the parameters of the lcpd are :

$$\beta_i, \quad \sigma_i^2, \quad i = 1, \dots, n. \quad (6)$$

Without loss of generality, we assume that the unconditional probability distribution has a zero mean. Extension to the more general case is trivial <sup>2</sup>.

### 2.2.3 Continuous variable with discrete and continuous parents

For a continuous variable  $X_i$ , with discrete and continuous parents  $\Pi_i$ , the lcpd is a table of conditional Gaussians. For each distinct configuration of the discrete parents, there is an associated conditional Gaussian where the mean of the Gaussian is a linear function of values of continuous parents.

$$P(X_i|\Pi_i) = \mathcal{N}(\beta_{ij}^T \Pi_i^{cont.}, \sigma_{ij}^2) \quad \begin{array}{l} i = 1, \dots, n \\ j = 1, \dots, q_i. \end{array} \quad (7)$$

Hence the parameters of the lcpd are:

$$\beta_{ij}, \quad \sigma_{ij}^2, \quad \begin{array}{l} i = 1, \dots, n \\ j = 1, \dots, q_i. \end{array} \quad (8)$$

## 3 Inference Engine

An important issue in BNs is computation of posterior probabilities of some variables given observation. The direct approach to compute and update the probabilities is not practical for a system with many variables. However, these operations within small sets of variables are quite easy. Using the factorization property of Bayesian networks, the inference problem can be divided into operations within smaller sets of variables (i.e. cliques).

The inference problem in general BNs is still a hot research topic. Several researchers have developed exact and approximate inference algorithms for different distributions. A review can be found in [16]. The most commonly used exact inference algorithm for discrete BNs is known as the JLO algorithm [19]. The algorithm can also be used for continuous Gaussian BNs [8]. In the following we will describe the discrete version of the algorithm. In the task of speech recognition we will consider hybrid networks where the continuous variables are fully observable. Hence the continuous nodes can also be considered as discrete while inferring probabilities. Therefore, the discrete inference engine is sufficient for our purposes.

The JLO algorithm is a recursive message passing algorithm that works on the junction tree of the BN. The junction tree is constructed from the DAG

---

<sup>2</sup>Redefine  $\beta_i = [1 \quad \beta_i]^T$  and  $\Pi_i = [\mu_i \quad \Pi_i]^T$ .

using some graph-theoretic tools. Starting with a factorization of the JPD on the junction tree the message passing algorithm extends the local calculations in the cliques to the complete set of variables. In the sequel we describe an algorithm for constructing the junction tree and show how the message passing algorithm can be used for inference

### 3.1 Constructing the Junction Tree

In order to define the junction tree we need to define the term clique. A clique is a complete set of nodes which is not a proper subset of another complete set. A set of nodes is said to be complete if every pair of nodes in the set is linked. A junction tree is a tree of cliques that satisfies the running intersection property (RIP). RIP implies that if a node is contained in any 2 cliques, then it is contained in all the cliques in the unique path between them.

The first step in the construction of the junction tree is to obtain an equivalent undirected graph satisfying the associated Markov property of the BN. This is performed by the process of moralization. The moral graph of a DAG is obtained by introducing additional undirected edges between any two nodes with a common child and subsequently replacing all directed edges with undirected ones. The moralization process guarantees that a family of nodes (a node with all of its parents) will occur together in a clique.

The second step is to add sufficient edges to the moral graph to obtain a triangulated (chordal) graph. The aim of triangulation is to obtain a decomposable model such that the joint probability distribution can be factorized over the clique potential functions. An undirected graph is triangulated if every loop of length four or more has at least one chord. There are several ways to add a chord in a loop with length greater than four. Hence, the triangulation process is not unique. In general it is desired to obtain a triangulation with a minimum number of additional edges. However, this problem is NP-complete [24]. We use the Maximum Cardinality Search Fill-In algorithm to obtain a triangulation of a given undirected graph [5]. This algorithm can be implemented in linear time  $O(N + l)$ , where  $N$  is the number of nodes and  $l$  is the number of links in the graph [23]. Given an initial node the algorithm constructs a triangulated graph and returns an ordering of the nodes.

**Algorithm: Triangulate**

- Input : An undirected graph  $G = (X, L)$  with  $N$  nodes and an initial node  $X_i$ .
- Outputs : A fill-in  $L'$ , such that  $G' = (X, L \cup L)'$  is a triangulated graph. An ordering (numbering)  $\alpha(j)$  of nodes
  1. Set  $L' = \emptyset$ .
  2. Set  $j = 1$ , and  $\alpha(j) = X_i$ .
  3. An unnumbered node  $X_k$  is with a maximum number of numbered neighbors is assigned label  $j$ ,  $\alpha(j) = X_k$ .
  4. If  $\{X_k \cup Neighbors(X_k)\} \cap \{\alpha(1) \dots \alpha(j-1)\}$  is not complete, add necessary links to make this set complete and go to Step 2; otherwise, go to Step 5.

5. If  $j = N$ , then stop; otherwise let  $j = j + 1$  and go to Step 3.

Now that the model is decomposable, the last step is to determine the cliques and to construct the tree. In constructing the clique tree we use the linear time algorithm defined in [25]. The algorithm proceeds in two steps. The first step is tree formation, where a clique is defined for each node with its lower numbered neighbors. Then, the cliques are linked to satisfy the RIP property.

**Algorithm : Tree Formation**

- Input : A triangulated graph  $G = (X, L)$  with a numbering  $\alpha(j)$  of nodes.
- Outputs : The junction tree with non-maximal cliques.
  1. Form a clique for each variable and its lower numbered neighbors.
  2. Order the cliques in increasing order according to the highest numbered node in each clique.
  3. For each clique  $C_i$  in increasing order
    - \* Identify the subset of its constituent variables that have occurred in the lower numbered cliques.
    - \* Find a lower numbered clique  $C_j$  that contains this subset.
    - \* Make  $C_i$  the parent of  $C_j$ .

The resulting cliques are not maximal. In the next step a tree reduction algorithm is used to obtain the maximal cliques and replace the smaller cliques with the maximal ones.

**Algorithm : Tree Reduction**

- Input : A junction tree of cliques created by Tree Formation algorithm.
- Outputs : The junction tree with maximal cliques.
  - repeat
    1. Identify a child that is a superset of its parent.
    2. Contract the edge between the two and replace the parent. by the child
  - until no child is a superset of its parent.

**3.2 Message passing**

The joint probability density of a set of variables can be represented as a product of clique potential functions on the associated junction tree.

$$P(\mathbf{X}) = \prod_{C \in \mathcal{C}} a_C(\mathbf{X}_C) \tag{9}$$

This factorization can be generalized using the separators in the junction tree. For each edge joining two cliques  $C_i, C_j$  in the junction tree a separator set  $S$  is defined as the intersection of the nodes in  $C_i$  and  $C_j$ . Associating a separator

potential function to each separator set, we have the following factorization of the joint probability density.

$$P(\mathbf{X}) = \frac{\prod_{C \in \mathcal{C}} a_C(\mathbf{X}_C)}{\prod_{S \in \mathcal{S}} b_S(\mathbf{X}_S)} \quad (10)$$

In the above equation, whenever the denominator is zero, the division is defined to be zero. With any valid initialization of clique and separator potential functions, at the end of the message passing algorithm every potential function becomes the marginal density for the relative set of variables. And at any step of the algorithm the potential functions constitute a valid factorization.

After the initialization, the local message passing scheme proceeds as follows. A *flow* from clique  $C_i$  to its neighbor  $C_j$  is defined through their separator  $S_k$ . First the separator potential  $b_{S_k}(\mathbf{X}_{S_k})$  is updated by marginalizing the clique potential over the variables that are in  $C_i$  but not in  $S_k$ .

$$b_{S_k}^*(\mathbf{X}_{S_k}) = \sum_{C_i \setminus S_k} a_{C_i}(\mathbf{X}_{C_i}) \quad (11)$$

Then the update factor of the separator is used to update the potential of the destination clique  $C_j$ .

$$\lambda_{S_k}(\mathbf{X}_{S_k}) = \frac{b_{S_k}^*(\mathbf{X}_{S_k})}{b_{S_k}(\mathbf{X}_{S_k})} \quad (12)$$

$$a_{C_j}^*(\mathbf{X}_{C_j}) = a_{C_j}(\mathbf{X}_{C_j}) \lambda_{S_k}(\mathbf{X}_{S_k}) \quad (13)$$

Note that the factorized representation of the joint distribution is still valid with the locally updated potentials. In order to distribute the information in each clique to the whole tree a two-phase propagation algorithm is used in general. Given a root clique in the tree, the *collection-phase* absorbs the flows starting from the leaves towards the root. Once all the flows are collected in the root, messages are sent towards the leaves in the *distribution-phase*. At the end of the algorithm at most two flows must have occurred along each edge.

### 3.3 Initializing the Junction Tree

Given the conditional probability distributions  $p(X_i | \Pi_i)$  of the variables  $X_i$ , the initialization of the junction tree can be performed as follows.

1. Assign each  $X_i$  to just one clique.
2. For each clique  $C$  that is assigned with at least one variable define the potential function as the product of  $p(X_i | \Pi_i)$  over all  $X_i$  assigned to  $C$ .
3. For all separators and the remaining cliques define the potential function to be 1.

With the above initialization of the junction tree, the message passing algorithm yields a prior probability distribution in each clique and separator.

$$P(\mathbf{X}) = \frac{\prod_{C \in \mathcal{C}} P_C(\mathbf{X}_C)}{\prod_{S \in \mathcal{S}} P_S(\mathbf{X}_S)} \quad (14)$$

The same algorithm can also be used to compute posterior probability distributions given an observation of a subset of variables. Once the clique tree is properly initialized with the observation, the two-phase propagation algorithm diffuses the observation and the resultant potentials are the posterior marginal distributions in each clique and separator. The observations are used in the algorithm as follows.

Let  $\mathbf{X}_h$  be the subset of variables that are hidden or unobserved and let  $D_l = \{X_i = x_i^*, X_j = x_j^*, \dots\}$  be an observation of the set of variables  $\mathbf{X}_o = \mathbf{X} \setminus \mathbf{X}_h$ . In order to compute  $p(\mathbf{X}_h | D_l)$  we first define an evidence function such that

$$g(x_i) = \begin{cases} 1 & \text{if } x_i = x_i^* \\ 0 & \text{otherwise.} \end{cases} \quad (15)$$

After initializing the junction tree with the conditional probability distributions, we multiply each clique potential with the evidence function according to the variable assignment in the initialization step.

$$a_C(\mathbf{X}_C) = a_C(\mathbf{X}_C) * \prod_{i: X_i \in \mathbf{X}_C} g(x_i) \quad (16)$$

With the incorporation of evidence, when the propagation algorithm reaches equilibrium, the clique potentials are the joint probability of the local hidden variables and the observed evidence.

$$a_C(\mathbf{X}_C) = P(\mathbf{X}_C^h, D_l) \quad (17)$$

If the potential function at a clique is normalized to sum to 1, one gets the conditional probability of the local hidden variables given the observation,

$$P(\mathbf{X}_C^h | D_l). \quad (18)$$

If the clique potential is marginalized over the hidden variables the probability of the observed evidence is obtained.

$$P(D_l) = \sum_{\mathbf{X}_C^h} P(\mathbf{X}_C^h, D_l) \quad (19)$$

### 3.4 Generalized operations

Other than probability computations, the inference engine can also be used for finding the most probable configuration of the variables or selecting a random sample from the distribution.

#### 3.4.1 Most probable configuration

In order to find the most probable configuration of the variables with or without evidence, it is sufficient to replace the marginalization operation in the flows with max-marginalization. The max-margin of a potential  $a(X)$  on  $Y \subseteq X$  is defined as follows:

$$b(Y) = \max_{Y \setminus X} a(X) \quad (20)$$

When the propagation algorithm is run with max-marginalization incorporating some evidence, the best explanation of the evidence is obtained.

### 3.4.2 Random configuration

Another useful operation could be generation of random samples from the joint probability distribution. This is performed with a slight modification in the propagation algorithm. After the regular collection phase, the distribution phase is replaced by a sample random-configuration phase. Starting from the root clique, a random-configuration is picked from the distribution on the each clique. Whenever a random-configuration is picked in a clique, this is entered as evidence by setting all other entries to zero in the clique potential.

## 4 Learning parameters from data

In this section we handle the problem of learning parameters of a BN given a fixed structure and a set of observations. In the first section we describe the ML estimation algorithm. Next we assume that there is some prior knowledge of the parameters and show how to incorporate this information in MAP estimation. Finally we show how to handle incomplete data sets in the case of ML estimation and derive the EM update equations for different parameterizations.

### 4.1 Maximum Likelihood (ML) Estimation

In ML estimation, the parameters of the BN are chosen such that the likelihood of the observed data is maximum.

$$\hat{\Theta}_{ML} = \arg \max_{\Theta} p(D|S, \Theta) \quad (21)$$

ML estimation is a powerful method especially for large data sets. For small data sets, there are two basic problems about ML estimation, sparse data, and over-fitting [4], [2].

- *Sparse data* : Consider a discrete BN, and a dataset  $D$ . If there are no instances of a specific realization of a variable in  $D$ , then the sample likelihood does not exist. Hence, ML estimate is undefined. In order to define the ML estimate for  $n$  binary variables and a fully connected BN, one needs more than  $2^{k-1}$  instances in the data set [4].
- *Over-fitting* : For small data sets, the maximum likelihood value may be much greater than the actual value. This over-estimation decreases as the data set size increases and the likelihood converges to the actual value. However, even there is over-estimation, the ML solution attempts to fit the data as well as possible. This results in a set of parameters that fit to an insufficient data. This is called over-fitting. (see [4] for an illustrated explanation)

#### 4.1.1 Data likelihood

Let  $C_l$  denote a complete (i.e.  $C_l$  consists of the realization of all the variables) random sample from the jpd of  $\mathbf{X}$ . Using the factorization property of the jpd, the likelihood of  $C_l$  can be expressed as a product of local likelihoods.

$$P(C_l|S, \Theta) = \prod_{i=1}^n P(x_i|\pi_i, S, \Theta) \quad (22)$$

$x_i^k$  and  $\pi_i^j$  denote the respective realizations of  $X_i$  and  $\Pi_i$  in  $C_l$ . For a data set  $D$  consisting of  $L$  i.i.d. samples, the likelihood can be computed as:

$$P(D|S, \Theta) = \prod_{l=1}^L P(C_l|S, \Theta) \quad (23)$$

#### 4.1.2 ML parameter estimation for discrete BNs

For a discrete BN, the likelihood of a complete sample  $C_l$  is written as :

$$P(C_l|S, \Theta) = \prod_{i=1}^n P(x_i^k | \pi_i^j, S, \Theta) \quad (24)$$

$$= \prod_{i=1}^n \theta_{ijk} \quad (25)$$

$$= \prod_{i=1}^n \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{1_{ijk}} \quad (26)$$

$$(27)$$

$$1_{ijk} = \begin{cases} 1 & \text{if } X_i = x_i^k \text{ and } \Pi_i = \pi_i^j \text{ in } C_l \\ 0 & \text{otherwise} \end{cases} \quad (28)$$

The likelihood of a complete dataset  $D$ , where each random sample  $C_l$  is independent and identically distributed (i.i.d.), can be written as:

$$P(D|S, \Theta) = \prod_{l=1}^L P(C_l|S, \Theta) \quad (29)$$

$$= \prod_{i=1}^n \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}} \quad (30)$$

$$N_{ijk} = \sum_{l=1}^L 1_{ijk} \quad (31)$$

Maximizing the likelihood term is equivalent to maximizing the log likelihood. Hence, we obtain the ML parameter estimates as follows where  $N_{ijk}$  are computed by counts in the observed data.

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{\sum_{k=1}^{r_i} N_{ijk}} \quad \begin{matrix} i = 1, \dots, n \\ j = 1, \dots, q_i \\ k = 1, \dots, r_i. \end{matrix} \quad (32)$$

#### 4.1.3 ML parameter estimation for continuous BNs

For a continuous BN the likelihood of a complete dataset  $D$  is written as:

$$P(D|S, \Theta) = \prod_{l=1}^L P(C_l|S, \Theta) \quad (33)$$

$$= \prod_{l=1}^L \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp \left\{ -\frac{(x_i^l - \beta_i^T \pi_i^l)^2}{2\sigma_i^2} \right\} \quad (34)$$

Maximizing the log-likelihood , one obtains the following ML parameter estimates :

$$\hat{\beta}_i^T = \left[ \sum_{l=1}^L \pi_i^l \pi_i^{lT} \right]^{-1} \left[ \sum_{l=1}^L x_i^l \pi_i^{lT} \right] \quad (35)$$

$$\hat{\sigma}_i = \frac{\sum_{l=1}^L (x_i^l - \hat{\beta}_i^T \pi_i^l)^2}{L} \quad \text{for } i = 1, \dots, n \quad (36)$$

#### 4.1.4 ML parameter estimation for hybrid BNs

For hybrid BNs the likelihood can be decomposed according to the type (discrete or continuous) of the variable and its parents. Then, each factor in the decomposition can be maximized separately. Therefore, the ML parameters for discrete variables with discrete parents and continuous variables with continuous parents are as given above. For the parameters of continuous variables with discrete and continuous parents, maximization should be performed separately for each distinct configuration of discrete parents. Hence the estimates are obtained as:

$$\hat{\beta}_{ij}^T = \left[ \sum_{l=1}^L \pi_i^{jl} \pi_i^{jlT} \right]^{-1} \left[ \sum_{l=1}^L x_i^l \pi_i^{jlT} \right] \quad (37)$$

$$\hat{\sigma}_{ij} = \frac{\sum_{l=1}^L (x_i^l - \hat{\beta}_{ij}^T \pi_i^{jl})^2}{L} \quad \begin{array}{l} i = 1, \dots, n \\ j = 1, \dots, q_i \end{array} \quad (38)$$

## 4.2 Maximum a posteriori (MAP) estimation and Bayesian learning

The MAP estimate of a random variable  $\Theta$  is defined as the value that maximizes the a posteriori pdf given observed data  $D$ . Using Bayes rule, one can show that maximizing the a posteriori pdf is equivalent to maximizing the joint pdf of  $D$  and  $\Theta$ , which can be factorized as the product of the data likelihood and the a priori parameter pdf.

$$\hat{\Theta}_{MAP} = \arg \max_{\Theta} P(\Theta|D, S) \quad (39)$$

$$= \arg \max_{\Theta} P(D|\Theta, S)P(\Theta|S) \quad (40)$$

MAP estimation can be viewed as an approximation to Bayesian learning. In Bayesian approach, the data likelihood is computed by averaging over all possible parameter values (i.e. one takes the expectation using the prior parameter pdf).

$$P(D|S) = \int P(D|S, \Theta) \cdot P(\Theta|S) d\Theta \quad (41)$$

Hence, Bayesian parameter learning is the update of prior parameter pdf using the observed data.

$$P(\Theta|S, D) = \frac{P(\Theta|S)P(D|S, \Theta)}{P(D|S)} \quad (42)$$

MAP estimation approximates this integral with the maximum value of the integrand.

In order to use MAP estimation or Bayesian learning for the parameters of a BN, one needs to define the prior pdf for the parameters  $\Theta$ . An important point in the selection of the prior is to have a conjugate family so that the functional form remains the same in the presence of data (i.e. the prior and posterior pdfs have the same functional form). Construction of parameter priors for DAG nodes is discussed in [14].

The initialization of the priors is another issue, which requires prior knowledge over the domain. [18] defines a method to initialize parameter priors for BNs of discrete variables with multinomial distributions based on a set of assumptions. A similar method is extended to continuous case in [17].

### 4.3 EM algorithm for parameter learning in the case of incomplete data

In the case of incomplete data, the factorization property of the BN cannot be used to compute the data likelihood. The likelihood of an incomplete sample  $D_l$ , where  $\mathbf{X}_h$  is the set of unobserved (hidden) variables, turns out to be:

$$P(D_l|S, \Theta) = E\{P(D_l, \mathbf{X}_h|S, \Theta)\}. \quad (43)$$

The expectation is taken with respect to  $P(\mathbf{X}_h|S, \Theta)$ . The EM algorithm is based on forming the conditional expectation of the log-likelihood function for complete data, given the observed data [22].

$$Q(\Theta'|\Theta) = E_{\Theta'}\{\log P(D_l, \mathbf{X}_h|S, \Theta')\} \quad (44)$$

The algorithm starts with an initialization of the parameters  $\Theta$  and alternates between E-step and M-step. At the E-step,  $\Theta$  is fixed and  $Q$  is evaluated as a function of  $\Theta'$ . At the M-step,  $Q$  is maximized in  $\Theta'$ .

$$\hat{\Theta} = \arg \max_{\Theta'} Q(\Theta'|\Theta) \quad (45)$$

#### 4.3.1 EM algorithm for parameter learning in discrete BNs

Let  $C_l$  denote a completion of the incomplete sample  $D_l$  with some random realizations of  $\mathbf{X}_h$ . Then the auxiliary function is written as:

$$Q(\Theta'|\Theta) = \sum_{x_h} P(\mathbf{X}_h|D_l, S, \Theta) \log P(C_l|S, \Theta') \quad (46)$$

$$= \frac{1}{P(D_l|S, \Theta)} \sum_{i=1}^n \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \log \theta'_{ijk} \sum_{x_h} P(\mathbf{X}_h, D_l|S, \Theta) 1_{lijk} \quad (47)$$

$$= \frac{1}{P(D_l|S, \Theta)} \sum_{i=1}^n \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \log \theta'_{ijk} P(x_i^k, \pi_i^j|D_l, S, \Theta) \quad (48)$$

The M-step gives the parameters for the next iteration where we need to perform inference to compute the probabilities.

$$\hat{\theta}_{ijk} = \frac{P(x_i^k, \pi_i^j|D_l, S, \Theta)}{\sum_{k=1}^{r_i} P(x_i^k, \pi_i^j|D_l, S, \Theta)} \quad (49)$$

For a set of  $L$  observations, we have:

$$\hat{\theta}_{ijk} = \frac{\sum_{l=1}^L P(x_i^k, \pi_i^j | D_l, S, \Theta)}{\sum_{k=1}^{r_i} \sum_{l=1}^L P(x_i^k, \pi_i^j | D_l, S, \Theta)} \quad (50)$$

### 4.3.2 EM algorithm for parameter learning in continuous BNs

$$Q(\Theta' | \Theta) = \int P(\mathbf{X}_h | D_l, S, \Theta) \log P(C_i | S, \Theta') dx_h \quad (51)$$

$$= \int P(\mathbf{X}_h | D_l, S, \Theta) \sum_{i=1}^n \left[ \log \frac{1}{\sqrt{2\pi\sigma_i'^2}} - \frac{(x_i^l - \beta_i'^T \pi_i^l)^2}{2\sigma_i'^2} \right] dx_h \quad (52)$$

$$= \sum_{i=1}^n \left[ \log \frac{1}{\sqrt{2\pi\sigma_i'^2}} - \frac{1}{2\sigma_i'^2} \left[ E\{x_i^l\} - 2\beta_i'^T E\{x_i^l \pi_i^{lT}\} + E\{\pi_i^l \pi_i^{lT}\} \right] \right] \quad (53)$$

In the M-step the maximization over  $\beta_i'$  and  $\sigma_i'^2$  is performed for each  $i$  separately.

$$\hat{\beta}_i^T = \left[ E\{\pi_i^l \pi_i^{lT}\} \right]^{-1} \left[ E\{x_i^l \pi_i^{lT}\} \right] \quad (54)$$

$$\hat{\sigma}_i = E\{(x_i^l - \hat{\beta}_i^T \pi_i^l)^2\} \quad \text{for } i = 1, \dots, n \quad (55)$$

For a set of  $L$  observations, we have:

$$\hat{\beta}_i^T = \left[ \sum_{l=1}^L E\{\pi_i^l \pi_i^{lT}\} \right]^{-1} \left[ \sum_{l=1}^L E\{x_i^l \pi_i^{lT}\} \right] \quad (56)$$

$$\hat{\sigma}_i = \frac{\sum_{l=1}^L E\{(x_i^l - \hat{\beta}_i^T \pi_i^l)^2\}}{L} \quad \text{for } i = 1, \dots, n \quad (57)$$

The expectations are computed over  $P(\mathbf{X}_h | D_l, S, \Theta)$  which can be computed by inference using the initial parameter set.

### 4.3.3 EM algorithm for parameter learning in hybrid BNs

As in ML estimation, the parameter estimates of hybrid BNs are considered separately for different types of parents and children. The estimates for a hybrid configuration where a discrete node has both discrete and continuous parents, are obtained as follows:

$$\hat{\beta}_{ij}^T = \left[ \sum_{l=1}^L E\{\pi_i^{jl} \pi_i^{jlT}\} \right]^{-1} \left[ \sum_{l=1}^L E\{x_i^l \pi_i^{jlT}\} \right] \quad (58)$$

$$\hat{\sigma}_{ij} = \frac{\sum_{l=1}^L E\{(x_i^l - \hat{\beta}_{ij}^T \pi_i^{jl})^2\}}{L} \quad \begin{array}{l} i = 1, \dots, n \\ j = 1, \dots, q_i \end{array} \quad (59)$$

## 5 Learning structure from data

### 5.1 Structural learning with complete data

The structure of a BN, defines the conditional independencies among the variables. Learning structure means learning conditional independencies from observations. The parameters of a BN with a given structure are estimated by ML or MAP estimation. One can use a similar method for choosing a structure for the BN that fits to the observed data. However, structural learning is slightly different than parameter learning. If we consider ML estimation or MAP estimation with uninformative priors then the structure that maximizes the likelihood will be the result. In this case complex structures (i.e. that has more dependencies between variables) will be more capable to increase the likelihood because they have more degrees of freedom. However, the aim is to find a structure that encodes the independencies among the variables, that is we want our structure to be as simple as possible.

The general approach to structure learning is to introduce a scoring metric that evaluates each structure for the given data and then to search for the best structure according to this metric.

### 5.2 Scoring structure

The criteria for selecting a structure has two terms. One for maximizing the likelihood and one for minimizing the complexity.

$$Score(S, D) = \log P(D|\hat{\Theta}, S) - Pen(S) \quad (60)$$

An important property for a scoring metric is decomposability. If a metric is decomposable, it can be factorized as a multiplication of local metrics. The local metric is a function of a variable and its parents. Hence when comparing two structures, it is sufficient to compare the local metrics that differ.

#### 5.2.1 Bayesian Information Criteria (BIC), Minimum Description Length (MDL) Score

MDL says that, the best model of a collection of data items is the model that minimizes the sum of

- the length of the encoding of the model
- the length of the encoding of the data given the model

[21] uses the MDL principle to score a BN. The first term is related to the complexity of the model. The simpler the model the less the encoding length will be. Therefore the MDL scoring penalizes complex network structures. The second term is related to the likeliness of the data based on the model. In coding theory the codewords with higher probability are represented with shorter codes. Therefore the length of the encoding is a measure of likelihood of data. The shorter the encoding the more likely the data. Hence learning a BN based on the MDL score is equivalent to

- minimizing the penalty score for the complexity of the structure

- maximizing the likelihood of data given the structure, based on ML parameter estimates

BIC score results from the approximation of the likelihood of the data given the structure around the MAP or ML estimates of the parameters. The approximation results in the same formulation of MDL [16].

$$Score_{MDL,BIC} = \log P(D|\hat{\Theta}, S) - Dim(S) \quad (61)$$

- For a discrete BN

$$Dim(S) = \begin{cases} \sum_{i=1}^n [k_i \log_2 n + d(r_i - 1)q_i] & \text{for MDL} \\ \frac{\log L}{2} \sum_{i=1}^n (r_i - 1)q_i & \text{for BIC} \end{cases} \quad (62)$$

### 5.2.2 Bayesian Dirichlet (BD, BDe) Score

In Bayesian approach the data likelihood is computed by taking the expectation over all possible network structures.

$$P(D_l) = \sum_S P(D_l|S)P(S) \quad (63)$$

Bayesian structural learning is updating the belief on each structure by computing the posterior  $P(S|D)$ . Hence, Bayesian scoring metric is the posterior structure pdf, or equivalently  $P(S, D)$ .

For discrete BNs, [7] derived a Bayesian scoring metric based on the following assumptions.

- *Assumption 1 : Multinomial sample*

The network structure hypothesis is true if and only if the database  $D$  can be partitioned into a set of multinomial samples <sup>3</sup>.

- *Assumption 2 : Parameter Independence*

Given a network structure  $S$ , if  $P(S) > 0$ , then

$$\begin{aligned} - P(\Theta_S|S) &= \prod_{i=1}^n P(\Theta_i|S) \\ - P(\Theta_i|S) &= \prod_{j=1}^{q_i} P(\Theta_{ij}|S) \text{ for } i = 1, \dots, n \end{aligned}$$

With this assumption the likelihood  $p(D|S)$  can be factorized. The assumption of parameter independence corresponds to the assumption that one's knowledge is equivalent to having seen only *complete cases*.

- *Assumption 3 : Parameter modularity*

The parameters associated with a variable, only depends on its parents. This assumption provides the use of prior parameter distributions defined on a complete network for all other networks with the same parental configuration.

---

<sup>3</sup>Consider a one-variable domain  $U = y$  where  $y$  can take values  $y^1, \dots, y^r$ . A finite sequence of observations  $D = y_1, \dots, y_N$  is an  $r$ -dimensional multinomial sample if

- $D$  is exchangeable (interchanging any two observations in the sequence does not change the probability of  $D$ )
- $\theta_k > 0$  for  $k = 1, \dots, r$  and  $\sum_{k=1}^r \theta_k = 1$  and  $p(y_l = y^k | y_1, \dots, y_{l-1}, \theta) = \theta_k$

- *Assumption 4 : Dirichlet*

The parameters have a Dirichlet distribution.  $P(\Theta_{ij}|S) = Dir(\alpha_{ijk})$

This assumption is eliminated when only positive distributions are considered. It is proved that Dirichlet distribution is the only distribution that is consistent with previous assumptions [18].

- *Assumption 5 : Complete Data*

In this case the likelihood can be computed in closed form.

$$Score_{BD} = P(S) \cdot P(D|S) = \prod_{i=1}^n \prod_{j=1}^{q^i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})} \quad (64)$$

In [18], the BDe metric is derived with the following additional assumptions.

- *Assumption 6 : Hypotheses Equivalence*

$S$  denotes the same hypotheses for  $S^1$  and  $S^2$ , if they are isomorphic (i.e. they represent the same assertions of conditional independence).

With this property, all complete network structures can be considered under the same hypotheses.

- *Assumption 7 : Structure possibility*

All complete network structures have probability greater than zero.

$$P(S_c) > 0 \quad (65)$$

The additional assumptions imposes the following constraints on the Dirichlet exponents :

$$\alpha_{ijk} = \alpha_{sc} P(X_i = x_i^k, \Pi_i = \pi^j | S_c) \quad (66)$$

$\alpha_{sc}$  is the equivalent sample size that is a measure of the prior domain knowledge. It can be considered as the number of observations on the domain to achieve current knowledge. This constraint enables the initialization of the prior parameter pdf of an arbitrary network structure given the pdf of a complete structure.

The BDe metric also has the property of likelihood equivalence. That is, the data likelihoods for two structures  $S^1$  and  $S^2$  are the same if they are isomorphic. Hence, assuming uniform structure priors, the scores of two structures, that have the same conditional independence assertions, are the same.

The Bayesian metric for continuous BNs is handled in [17].

### 5.2.3 Structure Priors

In Bayesian scoring metrics the preference for the structure (i.e. to choose simpler structure) is represented in the prior structure distribution. The structure priors are defined so as to penalize the structures that deviate from the initial structure. In [18] the deviation is represented as the number of different arcs between two networks. For a single variable this is denoted as  $\delta_i$ .

$$P(S) = c \cdot \prod_{i=1}^n \kappa^{\delta_i} \quad 0 < \kappa < 1 \quad (67)$$

In [3], Buntine considers different penalty factors for different arcs.

### 5.3 Searching structure space

Searching the structure space for high scored structures is the next issue in structure learning. It has been shown in [6] that finding the structure with maximum scoring is NP-hard. Therefore for arbitrary structures, heuristic search algorithms are used.

- *K2 search* : Initialize with an ordering of nodes such that the parents of a node are listed above the node. Starting with an empty structure and tracing the node list in order, one adds a parent to the node that increases the score. The number of parents to be added to a single variable may be limited to a predetermined constant for fast inference [7].
- *Local search* : Initialize with a network structure, possibly random. Evaluate the change in the score for all arc changes on this network and choose the one that has the maximum change. Continue this process until no more arc changes increases the score. This algorithm generally sticks into local maxima.
- *Iterated hill-climbing* : Apply local search until local maximum. Randomly perturb the structure and repeat the process for some manageable number of iterations.
- *Simulated annealing* : Initialize the system with some temperature  $T_0$ . Evaluate

$$p = \exp\left\{\frac{\Delta e}{T_0}\right\}. \tag{68}$$

If  $p = 0$  make the change  $e$ ; otherwise make the change  $e$  with probability  $p$ . Repeat the evaluation and selection  $\alpha$  times or until  $\beta$  changes. If there are no changes is  $\alpha$  repetitions stop searching; otherwise lower the temperature multiplying by  $0 < \gamma < 1$ , and continue searching.

There are also some polynomial algorithms for some special cases (i.e. for structures where each node can have only one parent, see [6] for a brief review).

### 5.4 Structural EM algorithm with incomplete data

In the incomplete data case the parameters corresponding to a specific structure cannot be obtained without a parametric search algorithm. Hence, the scoring metrics described above, are not decomposable and cannot be evaluated directly. The structural EM algorithm evaluates the expected score of a network based on some initial network [11], [12].

$$Q(S', \Theta' | S, \Theta) = E_{S, \Theta}\{Score(S, D)\} \tag{69}$$

The expectation is taken with respect to  $P(\mathbf{X}_h | D, S, \Theta)$ . The computation of the expected score generally requires inference within the previous network  $(S, \Theta)$ .

The SEM algorithm starts with an initialization of the structure and corresponding parameters. At the E-step, the expected score is computed for the candidate structure. At the M-step, the search algorithm determines the next

candidate structure to maximize the expected score. These two steps alternate until the maximizing structure is found.

An alternative SEM algorithm is also proposed that performs parametric maximization for some specified number of steps within structural maximization [11]. This leads to computational advantages, since parametric search is computationally cheaper than structure search.

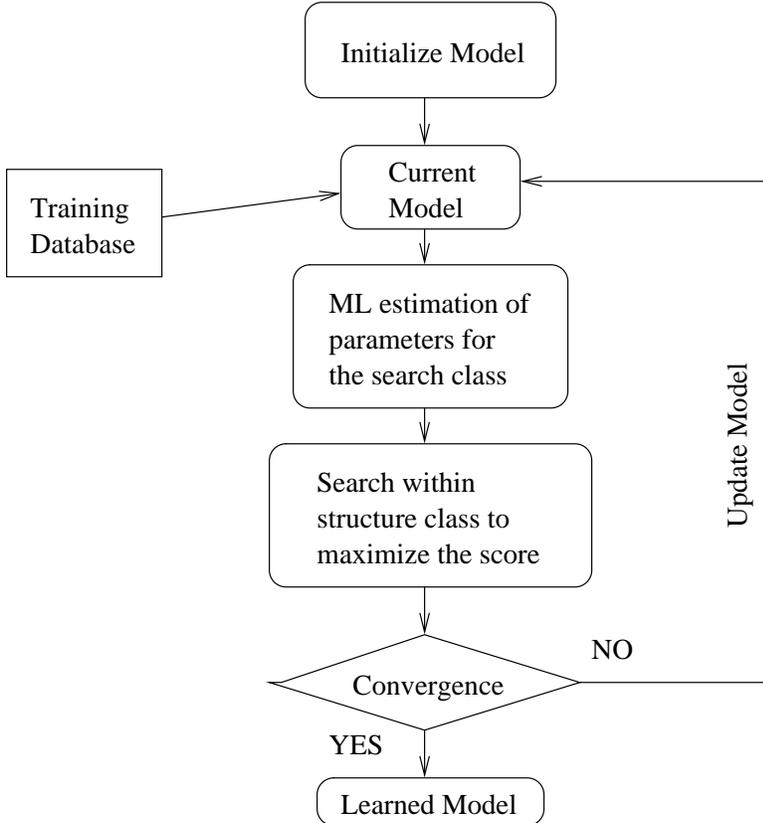


Figure 1: *SEM algorithm*

## 6 Dynamic Bayesian networks

### 6.1 Definition

A DBN encodes the joint probability distribution of a time-evolving set of variables  $X[t] = \{X_1[t], \dots, X_n[t]\}$ . If we consider  $T$  time slices of variables, the DBN can be considered as a (static) BN with  $T \times n$  variables. Using the factorization property of BNs, the joint probability density of  $\mathbf{X}_1^T = \{X[1], \dots, X[T]\}$  can be written as :

$$P(X[0], \dots, X[T]) = \prod_{t=1}^T \prod_{i=1}^n P(X_i[t] | \Pi_{it}) \quad (70)$$

where  $\Pi_{i_t}$  denotes the parents of  $X_i[t]$ .

## 6.2 Dependency range and structural inductions

In the BNs literature, DBNs are defined using the assumption that  $X[t]$  is Markovian and stationary [10] [9] [20] [15] [13]. The time dependency properties of  $X_i[t]$  determines the parents, and hence the network structure at time  $t$ . If the process is stationary, then the network structure is repeating for each time instant. However, care must be taken for the boundary regions depending on the dependency range (i.e. the initial time slice for a first order Markovian process).

More generally, violating the Markov assumption, one can assume that dependency is both to the past and to the future. More formally, we consider that the process  $X[t]$  satisfies:

$$P(X_i[t]|\mathbf{X}_1^{t+\tau_f}) = P(X_i[t]|X[t-\tau_p], \dots, X[t+\tau_f]) \quad (71)$$

for some integers  $\tau_p$  and  $\tau_f$ . Graphically, the above assumption states that a variable at time  $t$  can only have parents in the interval  $[t-\tau_p, t+\tau_f]$ . The boundaries are the first  $\tau_p$  and the last  $\tau_f$  time slices. The dependency time windows in these regions are different. This means that we do not have the repeating transition structure in these time slices.

Hence, the overall DBN structure can be decomposed with respect to three time intervals : In  $[1, \tau_p]$  and  $[T-\tau_f+1, T]$  we have  $\tau_p$  and  $\tau_f$  different structures, respectively. In  $[\tau_p+1, T-\tau_f]$  the structure is repeating. With this consideration, the DBN representation simplifies to  $(\tau_p + \tau_f + 1)$  static BNs:  $\tau_p$  initial networks,  $\tau_f$  final networks and a transition network. These networks are constructed considering a single time slice. Only the variables at time slice  $t$  can have parents in each of these networks. With this property, the jpd can be factorized over these networks.

$$P(X[0], \dots, X[T]) = \prod_{i=0}^n \left[ \prod_{t=0}^{\tau_p-1} P(X_i[t]|\Pi_i^I[t]) \prod_{t=\tau_p}^{T-\tau_f} P(X_i[t]|\Pi_i^T[t]) \prod_{t=T-\tau_f-1}^T P(X_i[t]|\Pi_i^F[t]) \right] \quad (72)$$

This is similar to the representation in [13], where first order Markov assumption is used, resulting in one initial network and a transition network.

### Example :

Consider a DBN for the variables  $X_o[t], X_h[t]$ , with the following assumptions :

$$P(X_h[t]|\mathbf{X}_1^{t+1}) = P(X_h[t]|X_h[t-2], X[t-1]) \quad (73)$$

$$P(X_o[t]|\mathbf{X}_1^{t+1}) = P(X_o[t]|X_h[t-1], X[t], X[t+1]) \quad (74)$$

The DBN structure for 5 time slices is given in Figure 2. The maximum past and future dependency parameters are,  $\tau_p = 2$  and  $\tau_f = 1$ . Therefore, the DBN can be represented with 2 initial, 1 final and a transition static networks. The transition network can directly be constructed from the dependency assumptions as shown in Figure 4(c).

In constructing the initial and final networks we have to consider the dependency assumptions for each time slice in the boundaries. For  $t = 1$ , the assumptions for  $X_o[1]$  and  $X_h[1]$  can be restated as follows.

$$P(X_h[1]|\mathbf{X}_1^2) = P(X_h[1]) \quad (75)$$

$$P(X_o[1]|\mathbf{X}_1^2) = P(X_o[1]|X[1], X[2]) \quad (76)$$

It is easy to see that these assumptions imply the structure shown in Figure 3(a). Similarly considering the assumptions for  $t = 2$  and  $t = T$ , one obtains the network structures given in Figure 3(a) and 4(d), respectively.

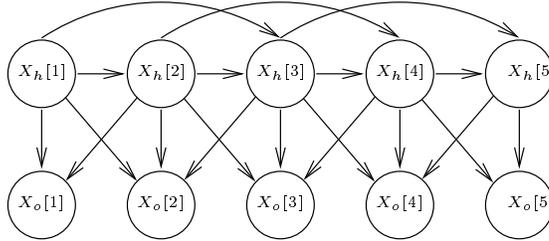


Figure 2: *DBN for  $T = 5$*

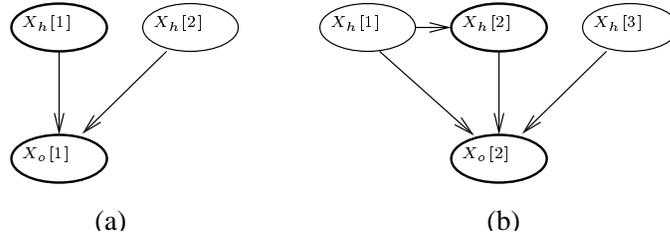


Figure 3: *Initial networks*

### 6.3 Inference and Learning in DBNs

A direct approach to infer probabilities in a DBN, is to construct a huge static BN for the desired time window of variables and then use the general inference algorithms for static BNs. This requires that the end of the time sequence is known. For more general cases, Kjærulff describes a computational scheme for dynamically updating the inference engine with incoming data [20].

Learning algorithms for BNs can also be used for DBNs. With the assumption of stationarity, the factorization of the DBN over static BNs enables us to apply these algorithms [13]. However, to learn the static networks the observation sequence should be properly feed to each network (i.e the transition network should be feed with the proper transitions in the data).

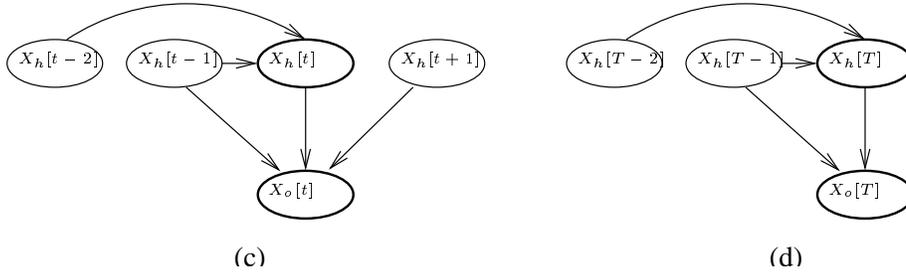


Figure 4: *Transition and final networks*

## 7 DBNs for speech recognition

In speech recognition each utterance is considered as a set of feature vectors from small time intervals that is assumed to be stationary. The general modelling strategy is to find a model for the production of these feature vectors in the sense that the likelihood of the observations based on the model is maximum. A commonly used model is the HMM model which is actually a DBN with two variables: a discrete hidden state variable and a continuous observed feature vector. The assumptions of HMM imposes a fixed structure for the DBN where

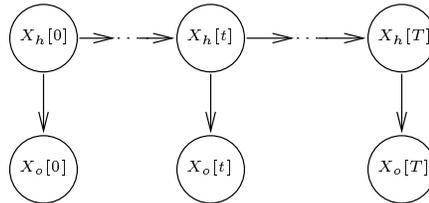


Figure 5: *HMM represented as a DBN*

each observed variable is the child of the hidden state in the same time slice and each hidden state variable is the child of the hidden state variable in the previous time slice. Although HMMs have shown significant performance in recognition, there are still some limitations. Considering the flexibility of Bayesian networks, a simple question is that why are we limited with the assumptions of first order HMMs.

In the DBN setting the inductions of HMMs can be extended. One can consider additional variables for each time slice to represent articulators in the speech production process [25]. Another approach is to impose additional dependencies among variables across time slices [1]. More importantly, the learning algorithms of BNs allows a totally data driven modelling. Not only the parameters, but also the dependency structure can be adapted to data.

In the following we investigate different DBN structures for speech recognition task, considering two variables for each time slice as in standard HMM. We assume that the hidden state variable is discrete with a multinomial distribution and the observed feature vectors have a conditional Gaussian density. Later we

use the set of possible structures to limit the search space of structural learning algorithms.

## 7.1 DBN structures for speech recognition

Our aim is to find the best dependency structure for the set of variables considered in the speech recognition task. Searching over all the possible DBN structures would be computationally infeasible. In this section, we propose to reduce this search to only “realistic” dependencies, in a physical and computational sense.

First, we do not consider direct dependencies between observable variables. We believe this is a reasonable assumption because, short-time correlations between cepstral coefficients are indirectly considered by  $\Delta$  and  $\Delta\Delta$  coefficients. Second, we do not allow future dependencies between the hidden variables. This is because past dependencies must be considered given causality of speech. Thus, given that the graph has to be acyclic, we cannot have both past and future dependencies between hidden variables. Finally, we do not allow a hidden variable to have an observable one as a parent. This is because there exists no exact algorithm to infer BNs where continuous variables have discrete children. We also assume that the hidden process is stationary. Therefore, the remaining authorized dependencies in the class of structures we are considering are the following :

Let  $X[t] = \{X_h[t], X_o[t]\}$  denote the hidden and observed variables at time  $t$  respectively, and  $\mathbf{X}_1^T = \{X[1], \dots, X[T]\}$  represent the set of the variables for a sequence of length  $T$ .

- A hidden variable at time  $t$  is independent of  $\mathbf{X}_1^{t-\kappa-1}$  given the last  $\kappa$  hidden variables, if  $t > \kappa$

$$P(X_h[t]|\mathbf{X}_1^{t-1}) = P(X_h[t]|X_h[t-\kappa], \dots, X_h[t-1]) \quad (77)$$

- The observation variable at time  $t$  is independent of all the other variables given the hidden variables in the time window  $[t-\tau_p, t+\tau_f]$ , for some integers  $\tau_p$  and  $\tau_f$

$$P(X_o[t]|\mathbf{X}_1^T \setminus \{X_o[t]\}) = P(X_o[t]|X_h[t-\tau_p], \dots, X_h[t+\tau_f]) \quad (78)$$

The stationarity assumption results in a repeating structure. Therefore, the triple  $(\kappa, \tau_p, \tau_f)$  determines completely the DBN structure. When  $(\kappa, \tau_p, \tau_f) = (1, 0, 0)$ , the model reduces to the standard first-order HMM (Figure 5) where Eq.(77) defines the state transition probabilities and Eq.(78) defines the observation probabilities. When  $(\kappa, \tau_p, \tau_f) = (2, 1, 1)$ , we have the structure of Figure 6. Hence, the search class of possible DBN structures is defined by the triples  $(\kappa, \tau_p, \tau_f)$  for,  $1 \leq \kappa \leq \kappa_{max}$ ,  $0 \leq \tau_p \leq \tau_{p_{max}}$ ,  $0 \leq \tau_f \leq \tau_{f_{max}}$ .  $(\kappa_{max}, \tau_{p_{max}}, \tau_{f_{max}})$  is an upper limit which defines the size of the search class.

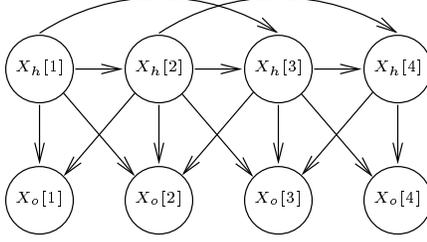


Figure 6: DBN structure with  $(\kappa, \tau_p, \tau_f) = (2, 1, 1)$ ,  $T = 4$

## 7.2 Learning DBNs for speech recognition

Now that we have defined the search class, the learning problem can be stated as follows:

Given a set of observations, find the “optimal” structure defined by the triple  $(\kappa, \tau_p, \tau_f)$ , and the associated conditional probabilities which best explain the data. The optimality will be considered in the sense that the likelihood of observations is maximum and the structural complexity is minimum.

In this section, we present the main lines of the learning algorithm for the class of DBN structures that we defined in the previous section. We assume that each discrete hidden variable takes  $M$  values and each observable variable has a Gaussian density which depend on the values of its parents. Therefore the numerical parameterization of our DBNs is the following:

$$\begin{aligned} P(X_h[t] = j | \Pi_h[t] = \mathbf{i}) &= a_{ij}[t], \quad \text{for } j = 1 \dots M. \\ P(X_o[t] | \Pi_o[t] = \mathbf{i}) &\sim \mathcal{N}(\mu_{\mathbf{i}}[t], \Sigma_{\mathbf{i}}[t]) \end{aligned} \quad (79)$$

The (possibly) vector-index  $\mathbf{i}$  is over all possible values of the variable’s parents, which depend on the structure, i.e. on the triple  $(\kappa, \tau_p, \tau_f)$ . Given the stationarity assumption, the parameters of these conditional probabilities are constant over time, when the structure is self-repeating, i.e.,

$$\begin{aligned} a_{ij}[t] &= a_{ij}, \quad \text{for } t = \kappa + 1, \dots, T \\ \mu_{\mathbf{i}}[t] &= \mu_{\mathbf{i}}, \\ \Sigma_{\mathbf{i}}[t] &= \Sigma_{\mathbf{i}}, \quad \text{for } t = \tau_p + 1, \dots, T - \tau_f \end{aligned} \quad (80)$$

Given a fixed structure in our search class, we have to estimate the parameters relative to this structure which achieve the maximum likelihood of data. This is done using the classical EM algorithm. Based on the dependency assumptions, the factorization of the joint probability function is as follows.

$$\begin{aligned} P(\mathbf{X}_{\mathbf{h}}^T, \mathbf{X}_{\mathbf{o}}^T) &= \prod_{t=1}^{\kappa} P(X_h[t] | \Pi_h[t]) \prod_{t=\kappa+1}^T P(X_h[t] | \mathbf{X}_{\mathbf{h}}^{t-\kappa}) \\ &\prod_{t=1}^{\tau_p} P(X_o[t] | \Pi_o[t]) \prod_{t=\tau_p+1}^{T-\tau_f} P(X_o[t] | \mathbf{X}_{\mathbf{h}}^{t+\tau_f}) \prod_{t=T-\tau_f+1}^T P(X_o[t] | \Pi_o[t]) \end{aligned} \quad (81)$$

Using the standard steps for deriving the EM update equations and considering the above factorization property, the update equations for a set of  $L$  observation sequences  $O_l = \{\mathbf{o}_1^l, \dots, \mathbf{o}_{T_l}^l\}$ , each of length  $T_l$ , are obtained as follows:

$$\begin{aligned}
\xi(t, j, \mathbf{i}, l) &= P(X_h[t] = j, \Pi_h[t] = \mathbf{i} | O_l) \\
\gamma(t, \mathbf{i}, l) &= P(\Pi_o[t] = \mathbf{i} | O_l)
\end{aligned} \tag{82}$$

for  $\kappa + 1 \leq t \leq T_l$

$$a_{ij} = \frac{\sum_{l=1}^L \sum_{t=\kappa+1}^{T_l} \xi(t, j, \mathbf{i}, l)}{\sum_{l=1}^L \sum_{t=\kappa+1}^{T_l} \sum_{j=1}^M \xi(t, j, \mathbf{i}, l)} \tag{83}$$

for  $1 \leq t \leq \kappa$

$$a_{ij}[t] = \frac{\sum_{l=1}^L \xi(t, j, \mathbf{i}, l)}{\sum_{l=1}^L \sum_{j=1}^M \xi(t, j, \mathbf{i}, l)} \tag{84}$$

for  $\tau_p + 1 \leq t \leq T_l - \tau_f$

$$\begin{aligned}
\mu_{\mathbf{i}} &= \frac{\sum_{l=1}^L \sum_{t=\tau_p+1}^{T_l-\tau_f} \mathbf{o}_t^l \gamma(t, \mathbf{i}, l)}{\sum_{l=1}^L \sum_{t=\tau_p+1}^{T_l-\tau_f} \gamma(t, \mathbf{i}, l)} \\
\Sigma_{\mathbf{i}} &= \frac{\sum_{l=1}^L \sum_{t=\tau_p+1}^{T_l-\tau_f} \mathbf{o}_t^{l*} \mathbf{o}_t^l \gamma(t, \mathbf{i}, l)}{\sum_{l=1}^L \sum_{t=\tau_p+1}^{T_l-\tau_f} \gamma(t, \mathbf{i}, l)} - \hat{\mu}_{\mathbf{i}}^* \hat{\mu}_{\mathbf{i}}
\end{aligned} \tag{85}$$

for  $1 \leq t \leq \tau_p$  and  $T_l - \tau_f + 1 \leq t \leq T_l$

$$\begin{aligned}
\mu_{\mathbf{i}}[t] &= \frac{\sum_{l=1}^L \mathbf{o}_t^l \gamma(t, \mathbf{i}, l)}{\sum_{l=1}^L \gamma(t, \mathbf{i}, l)} \\
\Sigma_{\mathbf{i}}[t] &= \frac{\sum_{l=1}^L \mathbf{o}_t^{l*} \mathbf{o}_t^l \gamma(t, \mathbf{i}, l)}{\sum_{l=1}^L \gamma(t, \mathbf{i}, l)} - \hat{\mu}_{\mathbf{i}}[t]^* \hat{\mu}_{\mathbf{i}}[t]
\end{aligned} \tag{86}$$

Note that, the maximization of the auxiliary function should be handled for each time  $t$  in the initial and final networks. The computation of  $\xi(t, j, \mathbf{i}, l)$  and  $\gamma(t, \mathbf{i}, l)$  is an inference problem for DBNs, which can be handled easily using the JLO algorithm [19].

Now that we have the parameter learning algorithm for a given structure, the structural learning algorithm can be stated as follows:

- Specify the triple  $(\kappa_{max}, \tau_{p_{max}}, \tau_{f_{max}})$ , in order to define the search class.
- Run the structural EM algorithm. However, instead of using a random structure in the first iteration, we use the HMM structure. This way, we guaranty that the learned model will have a likelihood greater (or equal) than the HMM likelihood. In other words, the learned model is guaranteed to model speech with higher fidelity than HMMs.

The triple  $(\kappa_{max}, \tau_{p_{max}}, \tau_{f_{max}})$ , controls the size of the set of allowed structures. Therefore, a trade off can be made at this level over the complexity of the learning algorithm and the fidelity of the resulting model.

An important point to note is that the structural search and parameter updating for different structures is performed in the training phase only. Once

a specific model is learned for the given data set, the recognition is performed using an inference algorithm on the learned structure. In other words, all the computational effort is done during training: which is a major technical advantage. Thus, the only measure of the computational cost of our approach is given by the complexity of the inference algorithm used in decoding.

### 7.3 Topology representation in DBNs

In HMM speech recognition techniques, an important consideration that effects the performance is the topology of the model. In HMM literature the topology is encoded in the state transition matrix. A full transition matrix is used for an ergodic model where any transition is probable. In DBN setting the topology is encoded in the conditional probability table of the hidden state variable, which is identical to the state transition matrix when a 1<sup>st</sup> order state dependence assumption is used. We use the *left-to-right* topology which induces an upper triangular state transition matrix for 1<sup>st</sup> order HMMs.

Assume that the hidden state variables of a DBN model can take  $M$  different values,  $\{x_h^1, \dots, x_h^M\}$ . Then for a left-to-right topology, the state transition probabilities has the following properties:

$$P(X_h[0] = x_h^i) = \begin{cases} 0, & \text{if } i \neq 1 \\ 1, & \text{if } i = 1 \end{cases} \quad (87)$$

$$P(X_h[t] = x_h^j | X_h[t-1] = x_h^i) = 0 \quad \text{if } j \neq i+1 \quad (88)$$

The first property ensures that the state sequence starts with the initial state. The second property implies that the state transitions occur from left to right and, no jumps more than one state is allowed.

In addition to using left-to-right transitions, an other desirable property of the state sequence could be imposed on the final state. In order to distribute the observation sequence (of length  $T$ ) to all states of the model, one could force the state sequence to end at the last state  $x_h^M$ . This could be achieved by imposing the following property on the conditional probability table of the hidden state variable at time  $T$ .

$$P(X_h[T] = x_h^M | X_h[T-1] = x_h^i) = 0 \quad \text{if } i \neq M, M-1 \quad (89)$$

## 8 Experiments

In this section, we provide a preliminary evaluation of the performance of our approach. Our primary goal here is to illustrate the potential of our approach. We are not trying to tune the parameters in order to achieve the best performances.

### 8.1 Isolated digit recognition

In isolated word recognition, we model each word in the database with a DBN and train the models from a given set of utterances. The structure and parameter learning of the model is performed using the techniques described in the previous sections. Once the models are trained for each word, in the test phase, we compute the likelihood of each test utterance on each model. The

recognition is based on maximum likelihood principle. The model that gives the maximum likelihood is chosen as the recognized word.

The experiments are carried out on the isolated part of the Tigits database in which 112 (resp. 113) speakers are used for training (resp. test). Each speaker uttered 11 digits twice. The size of each observation vector is 35, consisting of 11 MFCC (energy dropped), 12  $\Delta$ , and 12  $\Delta\Delta$ . Each hidden variable takes 4 values,  $M = 4$ . Each observable variable has a single Gaussian distribution with a diagonal covariance matrix. We assume a left-to-right transition topology for all the structures considered.

For the purpose of this experiment the upper-bound on the dependence parameters are chosen to be  $(\kappa_{max}, \tau_{p_{max}}, \tau_{f_{max}}) = (2, 1, 1)$ . As a first step we evaluate all possible structures for all digits using the BIC score. This experiment gives us the global maximum among the structure space for each digit. We observe that for all the digits, the structure with the highest score is the same<sup>4</sup> with the dependency parameters as,  $(\kappa, \tau_p, \tau_f) = (1, 0, 1)$ . This shows that future dependencies can be very important in modelling. The recognition rate obtained using this structure is 96.43%, while the rate given by the HMM structure is 92.86%. The improvement is remarkable given that the inference complexity of both structures is asymptotically the same. Indeed, due to the left-to-right topology assumption, the increase in the decoding complexity is only by a factor of 2 with respect to HMM decoding.

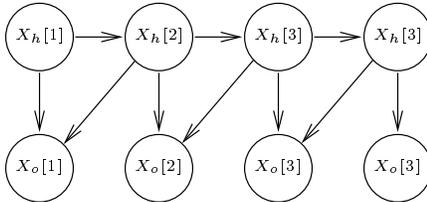


Figure 7: *The global maxima in the structure space of all the digits ( $T = 4$ ),  $(\kappa, \tau_p, \tau_f) = (1, 0, 1)$ .*

Next, we apply the structural learning algorithm using BIC scoring for each digit. The learned structure for '0', '2', '4', '5', '7', '9', is  $(\kappa, \tau_p, \tau_f) = (1, 0, 1)$ , and for '1', '3', '6', '8' it is  $(\kappa, \tau_p, \tau_f) = (1, 1, 0)$ . We observe a recognition rate of 96.63% with the learned models. For most of the digits, the SEM algorithm converges to the global maxima of the structure space. For the rest of the models, although the algorithm is struck in the local maxima the score is very close to the global maxima. Moreover, recognition results indicate that the modelling performance is acceptable.

It is also interesting to compare the recognition rates of all the structures within the search space. We performed separate parameter learning algorithms for each structure. In Table 1 we give a list of the recognition rates for each structure. The structures are listed in increasing structural complexity, where (1,0,1)-(1,1,0) and (2,0,1)-(2,1,0) have the same complexities respectively. One can see that, increasing model complexity generally increases the recognition

<sup>4</sup>We should note that the convergence to the same structure for all data sets is a special case. In general, it is possible to have a different structure for each data set.

performance. It should also be noted that none of the structures has a lower performance than the baseline HMM. MDL scoring leads to a good trade off between modeling fidelity and complexity. However we should observe that, although the structures (1,0,1) and (1,1,0) have the same complexity, and that (1,0,1) has a higher likelihood for the training set, the recognition performance is higher for (1,1,0). We relate this inconsistency to the imperfections in the data set.

$\kappa$	$\tau_p$	$\tau_f$	Recognition Rate
1	0	0	92.86
2	0	0	92.86
<b>1</b>	<b>0</b>	<b>1</b>	<b>96.43</b>
1	1	0	97.44
2	0	1	96.51
2	1	0	97.56
1	1	1	97.05
2	1	1	98.42

Table 1: *Recognition rate for different model structures (%)*.

## 9 Discussion and Future Work

We presented an acoustic modelling strategy based on Dynamic Bayesian Networks. The proposed modelling scheme involves both parameter and structural learning. This enables us to learn the CI assumptions from data as opposed to general HMMs where the CI are fixed. We tested the proposed scheme on an isolated digit recognition task based on separate digit models. We observe that the learned models give higher likelihood wrt. baseline HMMs indicating a higher modelling fidelity. Moreover, the MDL score is shown to be a successful metric in controlling the complexity of the learned model. However, one should note that the penalty term in the MDL/BIC score does not depend on the data. An improvement can be achieved by using a data driven penalty term.

As for the future work in order to increase the recognition score one can consider increasing the number of hidden states or modelling the output probability densities as mixtures of Gaussians. It has been shown that by modifying these parameters it is possible to achieve higher recognition rates in standard HMMs. Hence the same methodology can be applied to DBN models.

Another evaluation of different models with different dependencies can be performed by comparing them for equal number of parameters. (i.e adding dependencies in one model and increasing the number of mixtures in another model that would yield the same number of parameters). This kind of evaluation would reveal the importance of considering additional dependencies.

A more crucial step as a future work is to perform continuous recognition. In continuous recognition, the training is performed from a continuous database where the utterances are composed of sequences of words. Therefore one cannot train the models as in the isolated case. An initial processing must be performed to separate the words and/or phonemes in each utterance. The *embedded training* technique merges this process with parameter learning. This

technique assumes that the phoneme sequence of the utterance is given without specifying the transition instants and durations. The idea is to construct a super model that capsulates all the phonemes occurring in the utterance. Once this super model is trained with the given utterance, the parameters corresponding to each phone model is extracted. Besides individual phone model parameters, the trained model also learns the phoneme transition probabilities. These probabilities can be used as a bigram model.

In test phase, we construct a super model that includes all phoneme model parameters. The bigram model is also incorporated in this model. The values of the hidden state variable of the super model is the union of the states of each phoneme model. Hence the conditional probability table of the hidden state variables encode the transitions between phonemes as well as the phoneme state transitions. Decoding a given utterance using this model we obtain the most probable phoneme sequence. Another approach is to use multinets rather than a super model. In multinets, the dependencies of the network are changed according to a specific assignment of the variables. This allows us to consider simpler networks and hence reduce the computational cost.

Structure learning using embedded training is more complicated, since it is possible to have different structures for different phoneme models. In this case, the structure of the super model is not repeating. Moreover, it is not possible to construct the structure of the model without knowing the exact phoneme transitions. A possible approach to this problem is to start with the simple HMM structure, and perform embedded training for this model. Next we decode the training database using this simple model and find the phoneme transition instants. Then, the specific phoneme observations in each utterance can be used for structure learning for each phoneme separately as in the isolated case. However, the performance of the resulting system depends on the initial decoding of the HMM model. The errors will be transferred to the final system.

## References

- [1] Jeff A. Bilmes. *Natural Statistical Models for Automatic Speech Recognition*. PhD thesis, International Compute Science Institute, Berkeley, California, October 1999. ../bilmes/phd-bilmes.ps.
- [2] John Binder, Daphne Koller, Stuart Russell, and Keiji Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997. ../binder/mlj-apn.ps.
- [3] Wray Buntine. Theory refinement on bayesian networks. In *UAI'91*, pages 52–60, Los Angeles, 1991. ../buntine/buntine91.ps.
- [4] Wray Buntine. A guide to literature on learning probabilistic networks from data. *IEEE Trans. on Knowledge and Data Engineering*, 8:195–210, 1996. ../buntine/graphbib.pdf.
- [5] Enrique Castillo, José Manuel Gutiérrez”, and Ali S. Hadi. *Expert Systems And Probabilistic Network Models*. Springer-Verlag, New York, 1997.
- [6] David M. Chickering, Dan Geiger, and David Heckerman. Learning bayesian networks: Search methods and experimental results. In *Proc. AI and Statistics*, pages 112–128, 1995. ../chickering/aistat95.pdf.
- [7] Gregory F. Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.
- [8] Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Statistics for Engineering and Computer Science*. Springer, 1999.
- [9] Paul Dagum, Adam Galper, Eric Horvitz, and Adam Sevier. Uncertain reasoning and forecasting. *Int. Journal of Forecasting*, 11(1):73–87, March 1995. ../dagum/ijf.pdf.
- [10] Thomas Dean and Keiji Kanazawa. Probabilistic temporal reasoning. In *Seventh International Conference on Artificial Intelligence, AAAI-88*, pages 524–528, St. Paul, Minesota, 1988.
- [11] Nir Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Int. Conf. Machine Learning*, 1997. ../friedman/Fr1.ps.
- [12] Nir Friedman. The bayesian structural em algorithm. In *UAI-98*, 1998. ../friedman/Fr2.ps.
- [13] Nir Friedman, Kevin Murphy, and Stuart Russell. Learning the structure of dynamic probabilistic networks. In *UAI'98*, Madison, Wisconsin, 1998. ../friedman/uai98-dbn.ps.
- [14] Dan Geiger and David Heckerman. Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. Technical Report MSR-TR-98-67, Microsoft Research, Advanced Technology Division, 1999. ../geiger/tr-98-67.ps.

- [15] Z. Ghahramani. Learning dynamic bayesian networks, 1997. ../ghahramani/ghahramani97b.ps.
- [16] David Heckerman. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division, March 1995. ../heckerman/tr-95-06.ps.
- [17] David Heckerman and Dan Geiger. Learning bayesian networks. Technical Report MSR-TR-95-02, Microsoft Research, Advanced Technology Division, 1995. ../heckerman/tr-95-02.ps.
- [18] David Heckerman, Dan Geiger, and David. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995. ../heckerman/tr-94-09.ps.
- [19] F.V. Jensen, S.L. Lauritzen, and K.G. Olsen. Bayesian updating in recursive graphical models by local computations. *Computational Statistics and Data Analysis*, 4:269–282, 1990.
- [20] Uffe Kjaerulf. A computational scheme for reasoning in dynamic probabilistic networks. In *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 121–129, San Mateo, 1992. Morgan Kaufmann. ../kjaerulf/kjaerulf-92b.ps.
- [21] Wai Lam and Fahiem Bacchus. Learning bayesian belief networks an approach based on the mdl principle. *Computational Intelligence*, 10(4):269–293, 1994. ../lam/LBCL94.ps.
- [22] Steffen L. Lauritzen. The em algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.
- [23] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. *SIAM Journal of Computing*, 13:566–579, 1984.
- [24] M. Yannakakis. Computing the minimal fill-in is np-complete. *SIAM Journal of Algebraic Discrete Methods*, 2:77–79, 1981.
- [25] Geoffrey G. Zweig. *Speech Recognition with Dynamic Bayesian Networks*. PhD thesis, University of California, Berkeley, Spring 1998. ../zweig/phd-zweig.ps.