MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1611
C.B.C.L. Memo No. 151
June, 1997

Estimating Dependency Structure as a Hidden Variable

Marina Meilă **Michael I. Jordan** **Quaid Morris**

This publication can be retrieved by anonymous ftp to publications.ai.mit.edu.

Abstract

This paper introduces a probability model, the *mixture of trees* that can account for sparse, dynamically changing dependence relationships. We present a family of efficient algorithms that use EM and the Minimum Spanning Tree algorithm to find the ML and MAP mixture of trees for a variety of priors, including the Dirichlet and the MDL priors.

This report describes research done at the Dept. of Electrical Engineering and Computer Science, the Dept. of Brain and Cognitive Sciences, the Center for Biological and Computational Learning and the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Dept. of Defense and by the Office of Naval Research. Michael I. Jordan is a NSF Presidential Young Investigator. The authors can be reached at M.I.T., Center for Biological and Computational Learning, 45 Carleton St., Cambridge MA 02142, USA. E-mail: mmp@ai.mit.edu, jordan@psyche.mit.edu, quaid@ai.mit.edu.

1 INTRODUCTION

A fundamental feature of a good model is the ability to uncover and exploit independencies in the data it is presented with. For many commonly used models, such as neural nets and belief networks, the dependency structure encoded in the model is fixed, in the sense that it is not allowed to vary depending on actual values of the variables or with the current case. However, dependency structures that are conditional on values of variables abound in the world around us. Consider for example bitmaps of handwritten digits. They obviously contain many dependencies between pixels; however, the pattern of these dependencies will vary across digits. Imagine a medical database recording the body weight and other data for each patient. The body weight could be a function of age and height for a healthy person, but it would depend on other conditions if the patient suffered from a disease or were an athlete.

Models that are able to represent data conditioned dependencies are decision trees and mixture models, including the soft counterpart of the decision tree, the mixture of experts. Decision trees however can only represent certain patterns of dependecy, and in particular are not suited for representing sparse dependencies. Mixtures are more flexible and the rest of this paper will be focusing on one special case called *the mixture of spanning trees.*

We will consider domains where the observed variables are related by pairwise dependencies only and these dependencies are sparse enough to contain no cycles. Therefore they can be represented graphically as a *tree*. The structure of the dependencies may vary from one instance to the next. We index the set of possible dependency structures by a *structure variable* z (that can be observed or hidden) thereby obtaining a *mixture*.

In the framework of graphical probability models, tree distributions enjoy many properties that make them attractive as modelling tools: they have a flexible topology, are intuitively appealing, sampling and computing likelihoods are linear time, simple efficient algorithms for marginalizing and conditioning (quadratic or less in the dimension of the problem) exist. Fitting the best tree to a given distribution can be done exactly and efficiently. Trees can capture simple pairwise interactions between variables but they can prove insufficient for more complex distributions. Mixtures of trees enjoy most of the computational advantages of trees and, in addition, they are universal approximators over the space of all distributions. Therefore, they are appropriate as density estimators for domains where the dependency patterns become tree like when a possibly hidden variable is instantiated. But given the rich history of mixture models as classifiers they appear promising for these task as well.

Mixture models have been extensively used in the statistics and neural network literature. Of relevance to the present work are the mixtures of Gaussians, whose distribution space, in the case of continuous variables overlaps with the space of mixtures of trees. Mixtures of factorial distributions, a subclass of tree distributions, have been investigated recently by [8]. Work on fitting a tree to a distribution in a Maximum-Likelihood (ML) framework has been pioneered by Chow and Liu [1] and was extended to polytrees by Pearl [10] and to mixtures of trees with observed structure variable by Geiger [5] and Friedman [4].

This work presents efficient algorithms for learning mixture of trees models with unknown or hidden structure variable. The following section introduces the model; then, section 3 develops the basic algorithm for its estimation from data in the ML framework. Section 4 discusses the introduction of priors over mixtures of trees models and presents several realistic factorized and non-factorized priors for which the MAP estimate can be computed by modified versions of the basic algorithm. The properties of the model are verified by simulation in section 5 and section 6 concludes the paper.

2 THE MIXTURE OF TREES MODEL

In this section we will introduce the mixture of trees model and the notation that will be used throughout the paper. Let V denote the set of variables of interest. According to the graphical model paradigm, each variable is viewed as a vertex of a graph. Let r_v denote the number of values of variable $v \in V$, x_v a particular value of v, x_A an assignment to the variables in the subset A of V. To simplify notation x_V will be denoted by x.

We use trees as graphical representations for families of probability distributions over V that satisfy a common set of independence relationships encoded in the tree topology. In this representation, an edge of the tree shows a direct dependence, or, more precisely, the absence of an edge between two variables signifies that they are independent, conditioned on all the other variables in V. We shall call a graph that has no cycles a *tree*[1] and shall denote by E its edge set. A probability distribution T that is conformal with the tree (V, E) is a distribution that can be factorized as:

$$T(x) \;=\; \frac{\prod_{(u,v)\in E} T_{uv}(x_u, x_v)}{\prod_{v\in V} T_v(x_v)^{\deg v - 1}} \qquad (1)$$

Here $\deg v$ denotes the *degree* of v, e.g. the number of edges incident to node $v \in V$. The factors T_{uv} and T_v are the marginal distributions under T:

$$\begin{aligned} T_{uv}(x_u, x_v) &= \sum_{x_{V-\{u,v\}}} T(x_u, x_v, x_{V-\{u,v\}}) \\ T_v(x_v) &= \sum_{x_{V-\{v\}}} T(x_v, x_{V-\{v\}}). \end{aligned}$$

The distribution itself will be called a tree when no confusion is possible. If the tree is connected, e.g. it *spans* all the nodes in V, it is often called a *spanning tree.*

An equivalent representation for T in terms of conditional probabilities is

$$T(x) \;=\; \prod_{v\in V} T_{v|\mathrm{pa}(v)}(x_v | x_{\mathrm{pa}(v)}) \qquad (2)$$

[1] In the graph theory literature, our definition corresponds to a *forest*. The connected components of a forest are called trees.

The form (2) can be obtained from (1) by choosing an arbitrary root in each connected component and recursively substituting $\frac{T_{v\,\mathrm{pa}(v)}}{T_{\mathrm{pa}(v)}}$ by $T_{v|\mathrm{pa}(v)}$ starting from the root. pa(v) represents the parent of v in the thus directed tree or the empty set if v is the root of a connected component. The directed tree representation has the advantage of having independent parameters. The total number of free parameters in either representation is $\sum_{(u,v)\in E_T} r_u r_v - \sum_{v\in V}(\deg v - 1) r_v$.

Now we define a mixture of trees to be a distribution of the form

$$Q(x) \;=\; \sum_{k=1}^{m} \lambda_k T^k(x) \tag{3}$$

with

$$\lambda_k \geq 0,\; k = 1,\ldots,m; \qquad \sum_{k=1}^{m} \lambda_k \;=\; 1. \tag{4}$$

From the graphical models perspective, a mixture of trees can be viewed as a containing an unobserved choice variable z, taking value $k \in \{1,\ldots m\}$ with probability λ_k. Conditioned on the value of z the distribution of the visible variables x is a tree. The m trees may have different structures and different parameters. Note that because of the structure variable, a mixture of trees is not properly a belief network, but most of the results here owe to the belief network perspective.

3 THE BASIC ALGORITHM: ML FITTING OF MIXTURES OF TREES

This section will show how a mixture of trees can be fit to an observed dataset in the Maximum Likelihood paradigm via the EM algorithm [2]. The observations are denoted by $\{x^1, x^2, \ldots, x^N\}$; the corresponding values of the structure variable are $\{z^i,\, i = 1,\ldots N\}$.

Following a usual EM procedure for mixtures, the Expectation (E) step consists in estimating the posterior probability of each tree to generate datapoint x^i

$$Pr[y^i = k | x^{1,\ldots N}, model] \;=\; \gamma_k(i) \;=\; \frac{\lambda_k T^k(x^i)}{\sum_{k'} \lambda_{k'} T^{k'}(x^i)} \tag{5}$$

Then the expected complete log-likelihood to be maximized by the M step of the algorithm is

$$E[l_c \;|x^{1,\ldots N}, model] \;= \tag{6}$$

$$=\; \sum_{k=1}^{m} \Gamma_k [\log \lambda_k + \sum_{i=1}^{N} P^k(x^i) \log T^k(x^i)]$$

$$\Gamma_k \;=\; \sum_{i=1}^{N} \gamma_k(x^i), \quad k = 1,\ldots m \tag{7}$$

$$P^k(x^i) \;=\; \gamma_k(i)/\Gamma_k. \tag{8}$$

The maximizing values for the parameters λ are $\lambda_k^{new} = \Gamma_k/N$. To obtain the new distributions T^k, we have to maximize for each k the expression that is the negative of the crossentropy between P^k and T^k.

$$\sum_{i=1}^{N} P^k(x^i) \log T^k(x^i) \tag{9}$$

This problem can be solved exactly as shown in [1]. Here we will give a brief description of the procedure. First, one has to compute the mutual information between each pair of variables in V under the target distribution P

$$I_{uv} \;=\; \sum_{x_u x_v} P_{uv}(x_u, x_v) \log \frac{P_{uv}(x_u, x_v)}{P_u(x_u) P_v(x_v)}, \quad u, v \in V, u \neq v. \tag{10}$$

Second, the optimal tree structure E_T is found by a *Maximum Weight Spanning Tree* (MWST) algorithm using I_{uv} as the weight for edge $(u,v), \forall u, v \in V$. Once the tree is found, its marginals T_{uv} (or $T_{u|v}$), $(u,v) \in E_T$ are exactly equal to the corresponding marginals P_{uv} of the target distribution P. They are already computed as an intermediate step in the computation of the mutual informations I_{uv} (10).

In our case, the target distribution for T^k is represented by the posterior sample distribution P^k. Note that although each tree fit to P^k is optimal, for the encompassing problem of fitting a mixture of trees to a sample distribution only a local optimum is guaranteed to be reached. The algorithm is summarized in figure 1.

This procedure is based on one important assumption that should be made explicit now. It is the **Parameter independence assumption**: *The distribution $T^k_{v|\mathrm{pa}(v)}$ for any k, v and value of pa(v) is a multinomial with $r_v - 1$ free parameters that are independent of any other parameters of the mixture.*

Shared structure It is possible to constrain the m trees to share the same structure, thus constructing a truly Bayesian network. To achieve this, it is sufficient to replace the weights in step **M4** by $\sum_k I^k_{uv}$ and run the MWST algorithm only once to obtain the common structure E_T. The tree stuctures obtained by the basic algorithm are connected. The following section will give reasons and ways to obtain disconnected tree structures.

Missing variables are handled elegantly by trees. Any number of nonadjacent missing variables can be marginalized out in $\mathcal{O}(\max_v r_v)$ time and this bound grows exponentially with l, the size of the largest connected subset of missing variables.

Observed but unknown structure variable An interesting special case is the situation when the structure variable is in fact one of the observed variables (or a small subset thereof), but we don't know which one? To discover it, one can either: build several mixtures by conditioning on each one of the observables and then compare their posteriors, or: build one standard mixture model and then compare the mutual information between the structure variable and each of the others to identify the most likely candidate.

4 MAP MIXTURES OF TREES

In the previous section we have shown how to fit the ML mixture of spanning trees to a set of observations using the EM algorithm. Now we will extend the above procedure to the broader problem of finding the Maximum a Posteriori (MAP) probability mixture of trees for a given dataset. In other words, we will consider a nonuniform prior $P[model]$ and will be searching for the mixture of

Figure 1: **The Basic Algorithm: ML Fitting of a Mixture of Trees**

Input: Dataset $\{x^1, \dots x^N\}$
Initial model $m,\ T^k,\ \lambda^k,\ k = 1, \dots m$
Procedure MWST(weights) that fits a maximum weight spanning tree over V
Iterate until convergence:
E step: compute $\gamma_k^i,\ P^k(x^i)$ for $k = 1, \dots m,\ i = 1, \dots N$ by (5), (7), (8)
M step:
M1. $\lambda_k \leftarrow \Gamma_k/N,\ k = 1, \dots m$
M2. compute marginals $P_v^k,\ P_{uv}^k,\ u, v \in V,\ k = 1, \dots m$
M3. compute mutual information $I_{uv}^k u, v \in V,\ k = 1, \dots m$
M4. call MWST($\{I_{uv}^k\}$) to generate E_{T^k} for $k = 1, \dots m$
M5. $T_{uv}^k \leftarrow P_{uv}^k,\ ; T_v^k \leftarrow P_v^k$ for $(u, v) \in E_{T^k},\ k = 1, \dots m$

trees that maximizes

$$\log P[model|x^{1,\dots N}] \;=\; \log P[x^{1,\dots N}|model] + \log P[model] + \text{constant}. \quad (11)$$

Factorized priors The present maximization problem differs from the ML problem solved in the previous section only by the addition of the term $\log P[model]$. We can as well approach it from the EM point of view, by iteratively maximizing

$$E\left[\log P[model|x^{1,\dots N}, z^{1,\dots N}]\right] \;=\; \log P[model] + E[l_c(x^{1,\dots N}, z^{1,\dots N}|model)] \quad (12)$$

It is easy to see that the added term does not have any influence on the E step,which will proceed exactly as before. However, in the M step, we must be able to successfully maximize the r.h.s. of (12). In usual EM application this is enabled by the fact that we obtain a separate set of equations for the parameters of each mixture component. Therefore, we will look for priors over the trees parameters that also satisfy this decomposition. They are of the form

$$P[model] \;=\; P[\lambda_{1,\dots m}] \prod_{k=1}^{m} P[T_k] \quad (13)$$

This class of priors is in agreement with the parameter independence assumption and includes the conjugate prior for the multinomial distribution which is the Dirichlet prior. A Dirichlet prior over a tree can be represented as a table of fictitious marginal probabilities P'^k_{uv} for each pair u, v of variables plus an *equivalent sample size* N' that gives the strength of the prior [7]. It is now straightforward to maximize the a-posteriori probability of a tree: one has to replace the marginals P_{uv}^k in step **M2** by

$$\tilde{P}_{uv}^k \;=\; (NP_{uv}^k + N'P'^k_{uv})/(N + N'). \quad (14)$$

The Dirichlet prior implies the knowledge of detailed prior information about the model. In particular it implies that the number of mixture components m is known. When this is not the case, but there is information about the marginal relations between the variables one can introduce it in the form of one table of fictitious marginals and and an equivalent sample size N'. From it one can create a fictitious dataset of size N' to augment the true training set. Then, the training should proceed just like for an ordinary ML model fitting.

MDL (Minimum Description Length) priors are even less informative priors. They attempt to balance the number of parameters that are estimated with the amount of data available, usually by introducing a penalty on model complexity. For mixtures of trees one can proceed in two fashions, differing on whether they maintain or drop the parameter independence. First we will describe methods to reduce the number of parameters while keeping them independent.

Edge pruning and prior on m. To control the number of components m, one can introduce a prior $P[m]$ and compare model posteriors obtained from (11). To penalize the number of parameters in each component notice that adding a link (u, v) in a tree contributes $\Delta_{uv} = (r_u - 1)(r_v - 1)$ parameters w.r.t. a factorized distribution. One can also choose a uniform penalty $\Delta_{uv} = 1$. Introducing a prior $P[T] \;\propto\; \exp\left[-\beta \sum_{uv \in E_T} \Delta_{uv}\right]$ is equivalent to maximizing the following expression for each mixture component (the mixture index k being dropped for simplicity)

$$\operatorname*{argmax}_{E_T} \sum_{uv \in E_T} [\Gamma I_{uv} - \beta \Delta_{uv}] \;=\; \operatorname*{argmax}_{E_T} \sum_{uv \in E_T} W_{uv} \quad (15)$$

To achieve this for any choice of Δ_{uv} it suffices to replace the weights in step **M4** by W_{uv}^k and to modify the MWST procedure so as to consider only positive weight edges. This prior is a factorized prior as well[2]

Smoothing (or regularization) methods consider one comprehensive model class (full spanning trees and a sufficiently large m) and within it introduce a bias towards a small *effective number of parameters*. Here we discuss a few techniques that can be applied to trees and direct the reader to consult the vast existing literature related to smoothing in clustering and discrete probability estimation for futher information on this subject.

• **Penalizing the entropy of the structure variable** by introducing the penalty term $-\alpha H(\lambda_{1,\dots m})$. In this case, the λ_k cease being decoupled, and the resulting system of equations has to be solved numerically.

• **Smoothing with the marginal**. One computes the pairwise marginals for the whole dataset P_{uv}^{total} and replaces the marginals P_{uv}^k by

$$\tilde{P}_{uv}^k \;=\; (1 - \alpha)P_{uv}^k + \alpha P_{uv}^{total}, \quad 0 < \alpha < 1 \quad (16)$$

[2] Note that to use $P[m]$ together with edge pruning on has to compute the normalization constant in (11).

This method and several variations thereof are discussed in [9]. Its effect is to give a small probability weight to unseen instances and to draw the components closer to each other, thereby reducing the effective value of m. For the method to be effective in practice α is usually a function of Γ_k and P^k_{uv}.

5 EXPERIMENTAL RESULTS

We have tested our model and algorithms for their ability to retrieve the dependency structure in the data, as classifiers and as density estimators.

For the first objective, we sampled 30,000 datapoints from a mixture of 5 trees over 30 variables with $r_v = 4$ for all vertices. All the other parameters of the generating model and the initial points for the algorithm were picked at random. The results on retrieving the original trees were excellent: out of 10 trials, the algorithm failed to retrieve correctly only 1 tree in 1 trial. This bad result can be accounted for by sampling noise. The tree that wasn't recovered had a λ of only 0.02. Instead of recovering the missing tree, the algorithm fit two identical trees to the generating tree with the highest λ. The difference between the log likelihood of the samples of the generating model and the approximating model was 0.41 bits per example. On all the correctly recovered trees, the approximating mixture had a higher log likelihood for the sample set than the generating distribution.

We investigated the performance of mixtures of trees on two classification tasks from the UCI repository. For both tasks, we trained one model on the whole training set, treating the class variable like any ohter variable. In the testing phase, a new instance was classified by picking the most likely value of the class variable given the other variables settings.

The first task used the Glass database [6]. The data set has 214 instances of 9-dimensional continuous valued vectors. The class variable has 6 values. The continuous variables were discretized in 4 uniform bins each. We tested mixtures with different values of m, variable degrees of smoothing with the marginals and values of β for edge pruning. In smoothing with the marginals the coefficient α was inversely proportional to the marginal count $\Gamma_k P^k_{uv}$ for each marginal P^k_{uv}. For edge pruning we used a uniform penalty Δ_{uv}. For comparison we tried also mixtures of factorial distributions of different sizes. One seventh of the data, picked randomly at each trial, was used for testing and the rest for training. We replicate for comparison results obtained and cited in [8] on training/test sets of the same size. Table 2 shows a selection of the results we obtained. Smoothing with marginals proved to be bad for classification; therefore those results are not shown. The effect of edge pruning seems not to be significant on classification although, as expected, it increases the test set likelihood.

The second data set used was the Mushroom database [11]. This data set has 8124 instances of 23 discrete attributes (including the class variable, which is treated like any other attribute for the purpose of model learning). The training set comprised 6000 randomly chosen examples, and the test set was formed by the remaining 2124. The smoothing methods used were a) a penalty α_P on the entropy of the mixture variable and b) smoothing with the marginal according to (16) or similarly with a uniform distribution. The smoothing coefficient α_M was divided between the mixture components proportionally to $1/\Gamma_k$. For this dataset, smoothing was effective both in reducing overfitting and in improving classification performance. The results are shown in table 1. The soft classification colums expresses an intergrated measure of the confidence of the classifier. It is visible that besides the classification being correct, the classifier also has achieved high confidence.

We also tested the basic algorithm as density estimator by running it on a subset of binary vector representations of handwritten digits and measuring the compression rate. The datasets consists of normalized and quantized 8x8 binary images of handwritten digits made available by the US Postal Service Office for Advanced Technology. One dataset contained images of single digits in 64 dimensions, the second contained 128 dimensional vectors representing randomly paired digit images. The training, validation and test set contained 6000, 2000, and 5000 exemplars respectively. The data sets, the training conditions and the algorithms we compared with are described in [3]. We tried mixtures of 16, 32, 64 and 128 trees, fitted by the basic algorithm. The training set was used to fit the model parameters and the validation set to determine when EM has converged. The EM iteration was stopped after the first decrease in the log-likelihood on the validation set. For each of the two datasets we chose the mixture model with the highest log-likelihood on the validation set and using it we calculated the average log-likelihood over the test set (in bits per example). These results are shown in table 3. The other algorithms mentioned in the table are the mixture of factorial distributions (MF), the completely factorized model (which assumes that every variable is independent of all the others) called “Base rate”, the Helmholtz Machine trained by the wake-sleep algorithm [3] (Helmholtz Machine), the same Helmholtz Machine where a mean field approximation was used for training (Mean Field) and a fully visible and fully connected sigmoid belief network.

The results are very encouraging: the mixture of trees is the absolute winner for compressing the simple digits and comes in second as a model for pairs of digits. This suggests that our model (just like the mixture of factorized distributions) is able to perform good compression of the digit data but is unable to discover the independence in the double digit set. A comparison of particular interest is the comparison in performance between the mixture of trees and the mixture of factorized distribution. In spite of the structural similarities, the mixture of trees performs significantly better than the mixture of factorial distribution indicates that there exists some structure that is exploited by the mixture of spanning trees but can't be captured by a mixture of independent variable models.

6 CONCLUSIONS

This paper has shown a way of modeling and exploiting sparse dependency structure that is conditioned on

Table 1: Performance of mixture of trees models on the MUSHROOM dataset. m=10 for all models.

Algorithm	**Correctly classified**	**Soft class** $\sum \gamma_{\text{true}}/N$	**Test compression (bits/datapoint)**	**Train compression (bits/datapoint)**
No smoothing	.998	.997	27.63	27.09
Smooth w/ marginal $\alpha_M = 0.3,\ \alpha_P = 20$	1	.9999	27.03	26.97
Smooth w/ uniform $\alpha_M = 0.3,\ \alpha_P = 200$	1	.9997	27.39	27.02

Table 2: Performance of different algorithms on the GLASS dataset. MST is mixtures of spanning trees, MF is a mixture of factorial distributions.

Algorithm	**Classification performance**	**#runs**	**Algorithm**	**Classification performance**	**#runs**
MST m=30 β =0	*.82	10	MF m=35	.73	2
MST m=30 β =.02	.71	10	MF m=40	.63	1
MST m=30 β =.1	.70	1	MF m=48	*.80	8
MST m=30 β =1.	.67	2	MF from [8]	*.84	
MST m=35 β =0	.71	3	Flexible Bayes from [8]	.66	
MST m=40 β =.1	.73	3	C4 from [8]	.66	
MST m=25 β =0	.77	14	1Rw from [8]	.66	

Table 3: Compression rates (bits per digit) for the single digit (Digit) and double digit (Pairs) datasets. MST is mixtures of spanning trees, MF is a mixture of factorial distributions. A * marks the best performance on each dataset.

Algorithm	**Digits**	**Pairs**
gzip	44.3	89.2
Base Rate	59.2	118.4
MF	37.5	92.7
Mean Field	39.5	80.7
Helmholtz Machine	39.1	80.4
Fully Visible Bayes Net	35.9	*72.9
MST	***34.6**	**79.6**

values of the data. Without literally being a belief net, the mixture of trees that we introduced, by playing on variable topology independencies, is one in spirit. Trees do not suffer from the exponential computation demands that plague both inference and structure finding in wider classes of belief nets. The algorithms presented here are linear in m and N and quadratic in $|V|$. The loss in modelling power is compensated by using mixtures instead of single trees. The possibility of pruning a mixture of trees can play a role in classification, as a means of automatically selecting the variables that are relevant for the task.

The importance of using the right priors in constructing models for real-world problems can hardly be underestimated. In this context, two issues arise: 1. how is it possible to devise good priors over the class of mixtures of trees models ? and 2. what is the computational burden involved in taking priors into account ? The present paper has offered partial answers to both these questions: it has presented a broad class of priors that are efficiently handled in the framework of our algorithm and it has shown that this class includes important priors like the MDL prior and the Dirichlet prior.

ACKNOWLEDGEMENTS

Thanks to Brendan Frey for making the digits datasets available to us.

References

[1] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *"IEEE Transactions on Information Theory"*, IT-14(3):462–467, MAy 1968.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39:1–38, 1977.

[3] Brendan J. Frey, Geoffrey E. Hinton, and Peter Dayan. Does the wake-sleep algorithm produce good density estimators? In D. Touretsky, M. Mozer, and M. Hasselmo, editors, *Neural Information Processing Systems*, number 8, pages 661–667. MIT Press, 1996.

[4] Nir Friedman and Moses Goldszmidt. Building classifiers using Bayesian networks. In *Proceedings of the National Conference on Artificial Intelligence (AAAI 96)*, pages 1277–1284, Menlo Park, CA, 1996. AAAI Press.

[5] Dan Geiger. An entropy-based learning algorithm of bayesian conditional trees. In *Proceedings of the 8th Conference on Uncertainty in AI*, pages 92–97. Morgan Kaufmann Publishers, 1992.

[6] B. German. Glass identification database. U.C. Irvine Machine Learning Repository.

[7] David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learining*, 20(3):197–243, 1995.

[8] Petri Kontkanen, Petri Myllymaki, and Henry Tirri. Constructing bayesian finite mixture models by the EM algorithm. Technical Report C-1996-9, Univeristy of Helsinky, Department of Computer Science, 1996.

[9] Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, 8:1–38, 1994.

[10] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman Publishers, San Mateo, CA, 1988.

[11] Jeff Schlimmer. Mushroom database. U.C. Irvine Machine Learning Repository.