

Learning hybrid Bayesian networks from data

Stefano Monti[†]

[†]Intelligent Systems Program
University of Pittsburgh
901M CL, Pittsburgh, PA – 15260
smonti@isp.pitt.edu

Gregory F. Cooper[‡]

[‡] Center for Biomedical Informatics
University of Pittsburgh
8084 Forbes Tower, Pittsburgh, PA – 15261
gfc@cbmi.upmc.edu

Technical Report

ISSP-97-01

April 1997 (revised Aug '97)

Intelligent Systems Program
University of Pittsburgh
901 CL, Pittsburgh, PA - 15260

Abstract

We illustrate two different methodologies for learning *Hybrid Bayesian networks*, that is, Bayesian networks containing both continuous and discrete variables, from data. The two methodologies differ in the way of handling continuous data when learning the Bayesian network structure. The first methodology uses discretized data to learn the Bayesian network structure, and the original non-discretized data for the parameterization of the learned structure. The second methodology uses non-discretized data both to learn the Bayesian network structure and its parameterization. For the direct handling of continuous data, we propose the use of artificial neural networks as probability estimators, to be used as an integral part of the scoring metric defined to search the space of Bayesian network structures. With both methodologies, we assume the availability of a complete dataset, with no missing values or hidden variables.

We report experimental results aimed at comparing the two methodologies. These results provide evidence that learning with discretized data presents advantages both in terms of efficiency and in terms of accuracy of the learned models over the alternative approach of using non-discretized data.

1 Introduction

Bayesian belief networks (BNs), sometimes referred to as probabilistic networks, provide a powerful formalism for representing and reasoning under uncertainty. The construction of BNs with domain experts often is a difficult and time consuming task [16]. Knowledge acquisition from experts is difficult because the experts have problems in making their knowledge explicit. Furthermore, it is time consuming because the information needs to be collected manually. On the other hand, databases are becoming increasingly abundant in many areas. By exploiting databases, the construction time of BNs may be considerably decreased.

In most approaches to learning BN structures from data, simplifying assumptions are made to circumvent practical problems in the implementation of the theory. One common assumption is that all variables are discrete [7, 12, 13, 23], or that all variables are continuous and normally distributed [20]. We are interested in the task of learning BNs containing both continuous and discrete variables, drawn from a wide variety of probability distributions. We refer to these BNs as *Hybrid Bayesian networks*. The learning task consists of learning the BN structure, as well as its parameterization.

A straightforward solution to this task is to discretize the continuous variables, so as to be able to apply one of the well established techniques available for learning BNs containing discrete variables only. This approach has the appeal of being simple. However, discretization can in general generate spurious dependencies among the variables, especially if “local” discretization strategies (i.e., discretization strategies that do not consider the interaction between variables) are used¹. The alternative to discretization is the direct modeling of the continuous data as such. The experiments described in this paper use several real and synthetic databases to investigate whether the discretization of the data degrades structure learning and parameter estimation when using a Bayesian network representation.

The use of artificial neural networks (ANNs) as estimators of probability distributions presents a solution to the problem of modeling probabilistic relationships involving mixtures of continuous and discrete data. It is particularly attractive because it allows us to avoid making strong parametric assumptions about the nature of the probability distribution governing the relationships among the participating variables. They offer a very general *semi-parametric* technique for modeling both the probability mass of discrete variables and the probability density of continuous variables. On the other hand, as it was shown in the experimental evaluation in [28] (where only discrete data was used), and as it is confirmed by the evaluation reported in this paper, the main drawback of the use of ANN estimators is the computational cost associated with their training when used to learn the BN structure.

In this paper we continue the work initiated in [28], and further explore the use of ANNs as probability distribution estimators, to be used as an integral part of the scoring metric defined to search the space of BN structures. We perform an experimental evaluation aimed at comparing the new learning method with the simpler alternative of learning the BN structure based on discretized data. The results show that discretization is an efficient and accurate method of model selection when dealing with mixtures of continuous and discrete data.

The rest of the paper is organized as follows. In Section 2 we briefly introduce the Bayesian belief network formalism and some basics of how to learn BNs from data. In Section 3, we describe our learning method, and define the ANN-based scoring metric used to search the space of BN structures. In Section 4, we describe the use of artificial neural networks as probability distribution estimators. Finally, in Section 5 we present experimental results aimed at evaluating the efficacy of the proposed learning procedure, and at comparing it with a simple alternative based on the discretization of the continuous variables. We conclude the paper with a discussion of the results and with some suggestions for further research.

¹Most discretization techniques have been devised with the classification task in mind, and at best they take into consideration the interaction between the class variable and the feature variables individually. “Global” discretization for Bayesian networks learning, that is, discretization taking into consideration the interaction between all dependent variables, is a promising and largely unexplored topic of research, recently addressed in the work described in [19].

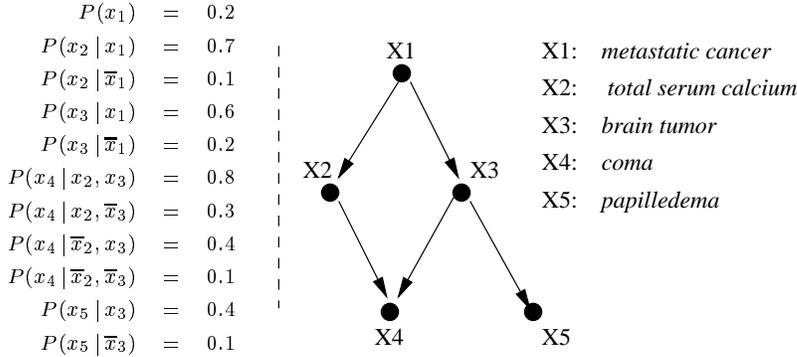


Figure 1: A simple belief network, with set of nodes $x = \{x_1, x_2, x_3, x_4, x_5\}$, and parent sets $\pi_{x_1} = \emptyset$, $\pi_{x_2} = \pi_{x_3} = \{x_1\}$, $\pi_{x_4} = \{x_2, x_3\}$, $\pi_{x_5} = \{x_3\}$. All the nodes represent binary variables, taking values from the domain $\{True, False\}$. We use the notation \bar{x}_i to denote $(x_i = False)$. The probability tables give the values of $p(x_i | \pi_{x_i})$ only, since $p(\bar{x}_i | \pi_{x_i}) = 1 - p(x_i | \pi_{x_i})$.

2 Background

A Bayesian belief network is defined by a triple (G, Ω, P) , where $G = (\mathcal{X}, E)$ is a directed acyclic graph with a set of nodes $\mathcal{X} = \{x_1, \dots, x_n\}$ representing domain variables, and with a set of arcs $E = \{(x_i, x_j) \mid x_i, x_j \in \mathcal{X}, x_i \neq x_j\}$ representing probabilistic dependencies among domain variables; Ω is the space of possible instantiations of the domain variables²; and P is a probability distribution over the instantiations in Ω . Given a node $x \in \mathcal{X}$, we use π_x to denote the set of parents of x in \mathcal{X} .

In Figure 1, we give an example of a simple Bayesian network structure, derived in part from [11]. By looking at the network structure, and by giving a causal interpretation to the links displayed, we see that *metastatic cancer* (x_1) is a cause of *brain tumor* (x_3), and that it can also cause an increase in *total serum calcium* (x_2). Furthermore, *brain tumor* can cause *papilledema* (x_5), and both *brain tumor* and an increase in *total serum calcium* can cause a patient to lapse into a *coma* (x_4).

The key feature of Bayesian networks is their explicit representation of conditional independence among events (domain variables). In particular, each variable is independent of its non-descendants given its parents. This property is usually referred to as the *Markov property*, and it allows for the parsimonious representation of the multivariate joint probability distribution over \mathcal{X} in terms of the univariate conditional distributions $P(x_i | \pi_i, \theta_i)$ of each variable x_i given its parents π_i , with θ_i the set of parameters needed to fully characterize the conditional probability. For example, with reference to the Bayesian network of Figure 1, where all the variables are discrete, each set of parameters θ_i is represented by means of a lookup table, with each entry in the table corresponding to the conditional probability $P(x'_i | \pi'_i, \theta_i)$ for a given instantiation of the variable x_i and its parents π_i .

The probability of any complete instantiation of \mathcal{X} can then be computed from the probabilities in the belief network. In fact, it has been shown [29, 35] that the joint probability of any particular instantiation of all n variables in a belief network can be calculated as follows:

$$P(x'_1, \dots, x'_n) = \prod_{i=1}^n P(x'_i | \pi'_{x_i}, \theta_i). \quad (1)$$

²An instantiation ω of all n variables in \mathcal{X} is an n -uple of values $\{x'_1, \dots, x'_n\}$ such that $x_i = x'_i$ for $i = 1 \dots n$.

2.1 Learning Bayesian belief networks³

In a Bayesian framework, ideally classification and prediction would be performed by taking a weighted average over the inferences of every possible BN containing the domain variables⁴. Since this approach is usually computationally infeasible, due to the large number of possible Bayesian networks, often an attempt has been made to select a high scoring Bayesian network for classification. We will assume this approach in the remainder of this paper.

The basic idea of the Bayesian approach is to maximize the probability $P(B_S | \mathcal{D}) = P(B_S, \mathcal{D})/P(\mathcal{D})$ of a network structure B_S given a database of cases \mathcal{D} . Because for all network structures the term $P(\mathcal{D})$ is the same, for the purpose of model selection it suffices to calculate $P(B_S, \mathcal{D})$ for all B_S .

So far, the Bayesian metrics studied in detail typically rely on the following assumptions: 1) given a BN structure, all cases in \mathcal{D} are drawn independently from the same distribution (*random sample* assumption); 2) there are no cases with missing values (*complete database* assumption; some more recent studies have relaxed this assumption [1, 8, 10, 21, 37]); 3) the parameters of the conditional probability distribution of each variable are independent (*global parameter independence* assumption); and 4) for discrete variables the parameters associated with each instantiation of the parents of a variable are independent (*local parameter independence* assumption). The last two assumptions can be restated more formally as follows. Let $\Theta_{B_S} = \{\theta_1, \dots, \theta_n\}$ be the complete set of parameters for the BN structure B_S , with each of the θ_i 's being the set of parameters that fully characterize the conditional probability $P(x_i | \pi_i)$. Also, when all the variables in π_i are discrete, let $\theta_i = \{\theta_{i1}, \dots, \theta_{iq_i}\}$, where θ_{ij} is the set of parameters defining a distribution that corresponds to the j -th of the q_i possible instantiations of the parents π_i . From Assumption 3 it follows that $P(\Theta_{B_S} | B_S) = \prod_i P(\theta_i | B_S)$, and from assumption 4 it follows that $P(\theta_i | B_S) = \prod_j P(\theta_{ij} | B_S)$ [36].

The application of these assumptions allows for the following factorization of the probability $P(B_S, \mathcal{D})$:

$$P(B_S, \mathcal{D}) = P(B_S)P(\mathcal{D} | B_S) = P(B_S) \prod_{i=1}^n s(x_i, \pi_i, \mathcal{D}), \quad (2)$$

where each $s(x_i, \pi_i, \mathcal{D})$ is a term measuring the contribution of x_i and its parents π_i to the overall score of the network structure B_S . The exact form of the terms $s(x_i, \pi_i, \mathcal{D})$ slightly differs in the Bayesian scoring metrics defined so far, and for the details we refer the interested reader to the relevant literature [7, 13, 23]. To date, closed-form expressions for $s(x_i, \pi_i, \mathcal{D})$ have been worked out for the cases when both x_i and π_i are discrete variables, or when both x_i and π_i are continuous (sets of) variables normally distributed; little work has been done in applying BN learning methods to domains not satisfying these constraints. Here, we only describe the metric for the discrete case defined by Cooper and Herskovits in [13], since it is the one we use in the experiments.

Given a Bayesian network B_S for a domain \mathcal{X} , let r_i be the number of states of variable x_i , and let $q_i = \prod_{x_s \in \pi_i} r_s$ be the number of possible instantiations of π_i . Let θ_{ijk} denote the multinomial parameter corresponding to the conditional probability $P(x_i = k | \pi_i = j)$, where j is used to index the instantiations of π_i , with $\theta_{ijk} > 0$, and $\sum_k \theta_{ijk} = 1$. Also, given the database \mathcal{D} , let N_{ijk} be the number of cases in the database where $x_i = k$ and $\pi_i = j$, and let $N_{ij} = \sum_k N_{ijk}$ be the number of cases in the database where $\pi_i = j$, irrespective of the state of x_i . Given the assumptions described above, and provided all the variables in \mathcal{X} are discrete, the probability $P(\mathcal{D}, B_S)$ for a given Bayesian network structure B_S is given by

$$P(\mathcal{D}, B_S) = P(B_S) \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(r_i)}{\Gamma(N_{ij} + r_i)} \prod_{k=1}^{r_i} \Gamma(N_{ijk}), \quad (3)$$

where Γ is the gamma function⁵.

³For a comprehensive guide to the literature on the subject, see [6].

⁴See the work described in [24, 25] for interesting applications of the *Bayesian model averaging* approach.

⁵Cooper and Herskovits [13] defined Equation (3) using factorials, although the generalization to gamma functions is straightforward.

Once a scoring metric is defined, a search for a high-scoring network structure can be carried out. This search task (in several forms) has been shown to be NP-hard [4, 9]. Various heuristics have been proposed to find network structures with a high score. One such heuristic is known as K2 [13], and it implements a greedy forward stepping search over the space of network structures. The algorithm assumes a given ordering on the variables. For simplicity, it also assumes a non-informative prior over parameters and structure. In particular, the prior probability distribution over the network structures is assumed to be uniform, and thus, it can be ignored in comparing network structures.

As previously stated, the Bayesian scoring metrics developed so far either assume discrete variables [7, 13, 23], or continuous variables normally distributed [20]. In the next section, we propose one generalization which allows for the inclusion of both discrete and continuous variables with arbitrary probability distributions.

3 An ANN-based scoring metric

In this section, we describe in detail the use of artificial neural networks as probability distribution estimators, to be used in the definition of a decomposable scoring metric for which no restrictive assumptions on the functional form of the class, or classes, of the probability distributions of the participating variables need to be made. The first three of the four assumptions described in the previous section are still needed. However, the use of ANN estimators allows for the elimination of the assumption of local parameter independence. In fact, the conditional probabilities corresponding to the different instantiations of the parents of a variable are represented by the same ANN, and they share the same network weights and the same training data. Furthermore, the use of ANNs allows for the seamless representation of probability functions containing both continuous and discrete variables.

Let us denote with $\mathcal{D}_l \equiv \{C_1, \dots, C_{l-1}\}$ the set of the first $l-1$ cases in the database, and with $x_i^{(l)}$ and $\pi_i^{(l)}$ the instantiations of x_i and π_i in the l -th case respectively. The joint probability $P(B_S, \mathcal{D})$ can be written as

$$\begin{aligned} P(B_S, \mathcal{D}) &= P(B_S)P(\mathcal{D} | B_S) = P(B_S) \prod_{l=1}^m P(C_l | \mathcal{D}_l, B_S) = \\ &= P(B_S) \prod_{l=1}^m \prod_{i=1}^n P(x_i^{(l)} | \pi_i^{(l)}, \mathcal{D}_l, B_S). \end{aligned} \quad (4)$$

If we assume uninformative priors, or decomposable priors on network structures, of the form $P(B_S) = \prod_i P(\pi_i)$, where $P(\pi_i)$ is the probability that π_i are the parents of x_i , the probability $P(B_S, \mathcal{D})$ is decomposable. In fact, we can interchange the two products in Equation 4, so as to obtain

$$P(B_S, \mathcal{D}) = \prod_{i=1}^n [P(\pi_i) \prod_{l=1}^m P(x_i^{(l)} | \pi_i^{(l)}, \mathcal{D}_l, B_S)] = \prod_{i=1}^n s(x_i, \pi_i, \mathcal{D}), \quad (5)$$

where $s(x_i, \pi_i, \mathcal{D})$ is the term between square brackets, and it is only a function of x_i and its parents in the network structure B_S (the prior $P(\pi_i)$ can be neglected if we assume a uniform prior over the network structures). The derivation illustrated in Equations 4 and 5 corresponds to the application of the prequential analysis discussed by Dawid in [14, 15]. Usually, the above decomposition is carried out in log terms, and it is interpreted as a predictive score, i.e., as a measure of the success of the model S in predicting the data \mathcal{D} . In fact, as it is shown more clearly in Equation 4, we form a predictive distribution of each case $C^{(l)}$ given the cases $\mathcal{D}_l = \{x^{(1)}, \dots, x^{(l-1)}\}$ already seen. Hence, the name of prequential analysis, which suggests sequential prediction. It corresponds to a theoretically sound form of cross-validation.

From a Bayesian perspective, each of the $P(x_i | \pi_i, \mathcal{D}_l, B_S)$ terms should be computed as follows:

$$P(x_i | \pi_i, \mathcal{D}_l, B_S) = \int_{\theta_i} P(x_i | \pi_i, \theta_i, B_S) P(\theta_i | \mathcal{D}_l, B_S) d\theta_i.$$

In most cases this integral does not have a closed-form solution; the following MAP approximation can be used instead:

$$P(x_i | \pi_i, \mathcal{D}_l, B_S) = P(x_i | \pi_i, \tilde{\theta}_i, B_S), \quad (6)$$

with $\tilde{\theta}_i$ the posterior mode of θ_i , i.e., $\tilde{\theta}_i = \operatorname{argmax}_{\theta_i} \{P(\theta_i | \mathcal{D}_l, B_S)\}$. As a further approximation, we use the maximum likelihood (ML) estimator $\hat{\theta}_i$ instead of the posterior mode $\tilde{\theta}_i$. The two quantities are actually equivalent if we assume a uniform prior probability for θ_i , and are asymptotically equivalent for any choice of positive prior. The approximation of Equation (6) corresponds to the application of the *plug-in* prequential approach discussed by Dawid [14].

Artificial neural networks can be designed to estimate $\hat{\theta}_i$ in both the discrete and the continuous case. Several schemes are available for training a neural network to approximate a given probability distribution, or density. In the next section, we describe the *softmax* model for discrete variables [5], and the *mixture density network* model introduced by Bishop in [2], for modeling conditional probability densities.

Notice that even if we adopt the ML approximation, the number of terms to be evaluated to calculate $P(\mathcal{D} | B_S)$ is still very large (mn terms, where m is the number of cases, or records, in the database, and n is the number of variables in \mathcal{X}), in most cases prohibitively so. The computation cost can be reduced by introducing a further approximation. Let $\hat{\theta}_i(l)$ be the ML estimator of θ_i with respect to the dataset \mathcal{D}_l . Instead of estimating a distinct $\hat{\theta}_i(l)$ for each $l = 1, \dots, m$, we can group consecutive cases in batches of cardinality t , and estimate a new $\hat{\theta}_i(l)$ for each addition of a new batch to the dataset \mathcal{D}_l rather than for each addition of a new case. Therefore, the same $\hat{\theta}_i(l)$, estimated with respect to the dataset \mathcal{D}_l , is used to compute each of the t terms $P(x_i^{(l)} | \pi_i^{(l)}, \hat{\theta}_i(l), B_S), \dots, P(x_i^{(l+t-1)} | \pi_i^{(l+t-1)}, \hat{\theta}_i(l), B_S)$.

With this approximation we implicitly make the assumption that, given our present belief about the value of each θ_i , at least t new cases are needed to revise this belief. We thus achieve a t -fold reduction in the computation needed, since we now need to estimate only m/t $\hat{\theta}_i$'s for each x_i , instead of the original m . In fact, application of this approximation to the computation of a given $s(x_i, \pi_i, \mathcal{D})$ yields:

$$\begin{aligned} s(x_i, \pi_i, \mathcal{D}) &= \prod_{l=1}^m P(x_i^{(l)} | \pi_i^{(l)}, \hat{\theta}_i(l), B_S) \\ &= \prod_{k=0}^{m/t-1} \prod_{l=tk+1}^{t(k+1)} P(x_i^{(l)} | \pi_i^{(l)}, \hat{\theta}_i(tk), B_S). \end{aligned} \quad (7)$$

With regard to the choice of an appropriate value for t , we can – for example – select a constant value for t , or we can choose to increment t as a function of $|\mathcal{D}_l|$. The second approach seems preferable. When estimating $\hat{\theta}_i(l)$, this estimate will be very sensitive to the addition of new cases when l is small, but it will become increasingly insensitive to the addition of new cases as l grows. For example, when $l = 1$, adding a new case to the training data set means doubling it, while if $l = 10,000$ an additional case in the data set is very unlikely to make a significant difference.

A scheme for the incremental updating of t can be summarized in the equation $t = \lceil \lambda l \rceil$, where l is the number of cases already seen (i.e., the cardinality of \mathcal{D}_l), and $0 < \lambda \leq 1$. For example, assuming we set $\lambda = 0.5$, given a data set of 50 cases, the updating scheme $t = \lceil 0.5l \rceil$ would require the training of the ANN estimators of $\hat{\theta}_i(l)$ for $l = 1, 2, 3, 5, 8, 12, 18, 27, 41$.

4 ANN probability estimators

In this section, we describe two models for the representation of conditional probability distributions with neural networks. These are the *softmax* model for discrete variables [5], and the *mixture density network* model introduced by Bishop in [2], for modeling conditional probability densities.

4.1 Softmax model for discrete variables

Let x_i be a discrete variable with r_i values and with a set of parents $\pi_i = \pi_i^c \cup \pi_i^d$, where π_i^c is the set of continuous parents, and π_i^d is the set of discrete parents. The conditional probability distribution $P(x_i | \pi_i)$ is approximated by a neural network with r_i output units, and r_{π_i} input units, where $r_{\pi_i} = |\pi_i^c| + \sum_{x_j \in \pi_i^d} (r_j - 1)$. The representation of a discrete input variable taking r_j values by means of $r_j - 1$ indicator variables is common practice in statistical regression. The r_i output units define the conditional probabilities $P(x_i = v_k | \pi_i)$, $k = 1, \dots, r_i$, as follows:

$$P(x_i = v_k | \pi_i) = \frac{e^{f_k(\pi_i)}}{\sum_{j=1}^{r_i} e^{f_j(\pi_i)}}, \quad (8)$$

where $f_k(\pi_i)$ is the linear output of the k -th output unit corresponding to the network input π_i . Notice that the probability $P(x_i = v_k | \pi_i)$ can be interpreted as the probability of class membership of π_i , i.e., as the probability that π_i belongs to the k -th of r_i classes. It has been proved that a neural network thus configured, with a sum-of-squares or cross-entropy error function, leads to network outputs that estimate Bayesian a posteriori probabilities of class membership [3, 32].

4.2 Mixture density networks for continuous variables

The approximation of probability distributions by means of finite mixture models is a well established technique, widely studied in the statistics literature [17, 38]. Bishop [2] describes a class of network models that combine a conventional neural network with a finite mixture model, so as to obtain a general tool for the representation of conditional probability distributions.

The probability $P(x_i | \pi_i, \mathcal{D}_l, B_S)$ can be approximated by a finite mixture of normals as is illustrated in the following equation (where we dropped the conditioning on \mathcal{D}_l and B_S for brevity):

$$P(x_i | \pi_i) = \sum_{k=1}^K \alpha_k(\pi_i) \phi_k(x_i | \pi_i), \quad (9)$$

where K is the number of mixture components, $0 \leq \alpha_k \leq 1$, $k = 1, \dots, K$, and $\sum_k \alpha_k = 1$, and where each of the kernel functions $\phi_k(x_i | \pi_i)$ is a normal density of the form:

$$\phi_k(x_i | \pi_i) = c \exp \left\{ -\frac{(x_i - \mu_k(\pi_i))^2}{2\sigma_k(\pi_i)^2} \right\} \quad (10)$$

with c the normalizing constant, and $\mu_k(\pi_i)$ and $\sigma_k(\pi_i)^2$ the conditional mean and variance respectively. The parameters $\alpha_k(\pi_i)$, $\mu_k(\pi_i)$, and $\sigma_k(\pi_i)^2$ can be considered as continuous functions of π_i . They can therefore be estimated by a properly configured neural network. Such a neural network will have three outputs for each of the K kernel functions in the mixture model, for a total of $3K$ outputs. The set of input units corresponds to the variables in π_i . It can be shown that a Gaussian mixture model such as the one given in Equation (9) can – with an adequate choice for K – approximate to an arbitrary level of accuracy any probability distribution. Therefore, the representation given by Equations (9) and (10) is completely general, and allows us to model arbitrary conditional distributions. More details on the mixture density network model can be found in [2].

Notice that the mixture density network model assumes a given number K of kernel components. In our case, this number is not given, and needs to be determined. The determination of the number of components of a mixture model is probably the most difficult step, and a completely general solution strategy is not available. Several strategies are proposed in [31, 33, 38]. However, most of the techniques are computationally expensive, and given our use of mixture models, minimizing the computational cost of the selection process becomes of paramount importance. Given a set of alternative model orders $\mathcal{K} = \{1, \dots, K^{\max}\}$, we consider two alternative strategies for the selection of the best order $K \in \mathcal{K}$: model selection based on a test set held

out during training (“hold-out”), and model selection based on the *Bayesian Information Criterion* (BIC) [30, 34].

Model selection by hold-out is performed by splitting the dataset \mathcal{D}_l into a training set $\mathcal{D}_l^{\text{train}}$ and a test set $\mathcal{D}_l^{\text{test}}$. A mixture density network \mathcal{M}_K is then trained on the training set $\mathcal{D}_l^{\text{train}}$ for each $K \in \mathcal{K}$, and the model order K that maximizes the probability $P(\mathcal{D}_l^{\text{test}} | \hat{\theta}, \mathcal{M}_K)$ is selected, where $\hat{\theta}$ is the ML estimator with respect to $\mathcal{D}_l^{\text{train}}$.

Model selection based on BIC aims at finding the model order K that maximizes $P(\mathcal{D}_l | \mathcal{M}_K)$. BIC provides the following asymptotic approximation:

$$P(\mathcal{D}_l | \mathcal{M}_K) = P(\mathcal{D}_l | \hat{\theta}, \mathcal{M}_K) - \frac{d}{2} \log l + O(1) \quad (11)$$

where d is the number of parameters in the model \mathcal{M}_K (in our case, the number of weights of the neural network), l is the size of the dataset, and $\hat{\theta}$ is the ML estimator of the parameters of \mathcal{M}_K (in our case, $\hat{\theta}$ is given by the outputs $\{\alpha_k(\pi_i), \mu_k(\pi_i), \sigma_k(\pi_i)^2 | k = 1, \dots, K\}$ of the trained neural network). Given certain regularity conditions, the error bound for Equation 11 is $O(1)$.

Since repeating the model order selection for each prequential term (i.e., for the estimation of each $\hat{\theta}_i(l)$) would be computationally too costly, we select the model order based on the whole dataset \mathcal{D} , and we then use the same selected order in each prequential term.

4.3 ANN training

For the training of the ANNs, we use the conjugate-gradient optimization algorithm described in [27]. This algorithm shows much faster convergence to the (possibly local) maximum than backpropagation. Currently, we do not use any regularization technique to control for over-fitting.

The number of hidden units is selected as a function of the number of inputs. More specifically, we set the number of hidden units to be half the number of input units, with a minimum of three hidden units. The ANN’s weights are randomly initialized with real values in the interval $(-.5, .5)$. This, however, is only true for the ANN corresponding to the first prequential term of the scoring metric. The ANNs corresponding to subsequent prequential terms have their weights initialized to the weights of the ANN for the previous prequential term. More specifically, for a given $s(x_i, \pi_i, \mathcal{D}_l)$ term, we need to train several ANNs, one for each of the prequential terms $P(x_i | \pi_i, \mathcal{D}_l, B_S)$, $l = 1, \dots, m$. Given the ANN trained on the database \mathcal{D}_l , we can use the weights of that network as the initialization of the weights for the ANN to be trained on the database \mathcal{D}_{l+1} (or \mathcal{D}_{l+t} if we use the updating scheme described at the end of Section 3). This strategy will be particularly beneficial for large sized \mathcal{D}_l , where the addition of a new case (or a few new cases) will not change significantly the estimated probability.

5 Experimental evaluation

In this section we describe the experimental evaluation we conducted to test the viability of use of the ANN-based scoring metric. We first describe the experimental design. We then present the results and discuss them.

5.1 Experimental Design

The experimental evaluation is primarily aimed at determining whether the new scoring metric, given by Equations (7), (8), and (9), which is applicable to continuous variables as well as to discrete variables, offers any advantage over the use of the scoring metric of Equation (3), which is applied to discretized data. To this end, we considered two learning algorithms:

- Algorithm A1 first performs a discretization of the data; then it searches for the highest scoring network structure based on the scoring metric of Equation (3); finally it estimates the parameters of the discovered structure by using ANN estimators applied to the original continuous data.
- Algorithm A2 searches for the highest scoring BN structure using the ANN-based scoring metric of Equation (7) applied to the original continuous data, and it also estimates the parameters of the discovered structure by means of ANN estimators.

We would expect the structure search by A1 to be faster than the structure search by A2, but possibly less accurate, due to the information loss resulting from discretization.

The discretization technique used with the algorithm A1 is a simple “constant density” discretization, whereby the value range of the continuous variable is partitioned into a given number of *bins* so that an approximately equal number of contiguous data points is assigned to each bin.

To make the comparison of the two algorithms simpler, the experiments were designed so as to compare the performance of the two algorithms in discovering the set of parents of a given response, or *class* variable. The evaluation is aimed at testing both the predictive accuracy of the two algorithms, and their capability of discovering relevant structural patterns in the data. With regard to the first goal, real data is fully appropriate, and it allows for a better testing of the robustness of the assumptions made. With regard to the second goal, simulated data generated from Bayesian networks whose structure and parameterization are known is more appropriate, since the generating BN represents the gold standard with which we can compare the model(s) selected by the learning procedure.

To assess the predictive accuracy of the two algorithms, we measured mean square error (MSE) and log score (LS) with respect to the class variable y on a test set distinct from the training set. The mean square error is computed with the formula:

$$\text{MSE} = \frac{1}{L} \sum_{\mathcal{D}^{\text{test}}} [y^{(l)} - \hat{y}(\pi_y^{(l)})]^2, \quad (12)$$

where $\mathcal{D}^{\text{test}}$ is a test set of cardinality L , $y^{(l)}$ is the value of y in the l -th case of $\mathcal{D}^{\text{test}}$, and $\hat{y}(\pi_y^{(l)})$ is the value of y predicted by the learned BN for the given instantiation $\pi_y^{(l)}$ of y 's parents. More specifically, $\hat{y}(\pi_y^{(l)})$ is the expectation of y with respect to the conditional probability $P(y | \pi_y^{(l)})$.

Similarly, the log-score LS is computed with the formula:

$$\text{LS} = -\log P(\mathcal{D}^{\text{test}}) = -\sum_{\mathcal{D}^{\text{test}}} \log P(y^{(l)} | \pi_y^{(l)}), \quad (13)$$

where $P(y^{(l)} | \pi_y^{(l)})$ is the conditional probability for y in the learned BN. With both MSE and LS the lower the score the better the model.

For the evaluation with real databases, we used databases from the data repository at UC Irvine [26]. In particular, we used the databases `AUTO-MPG` and `ABALONE`. These databases were selected because their class variable can be treated as a continuous variable. In the database `AUTO-MPG` the class variable is `miles-per-gallon`. In the database `ABALONE` the class variable is an integer proportional to the age of the mollusk, and can thus be treated as a continuous variable. The database `AUTO-MPG` has a total of 392 cases over eight variables, of which two variables are discrete, and six are continuous. The database `ABALONE` has a total of 4177 cases over nine variables, of which only one variable is discrete. All continuous variables were normalized.

Since we are only interested in selecting the set of parents of the response variable, the only relevant ordering of the variables needed for the search algorithm is the partial ordering that has the response variable as the successor of all the other variables.

All the statistics reported are computed over ten simulations. In each simulation, 10% of the cases is randomly selected as the test set, and the learning algorithms use the remaining 90% of the cases for training. Notice

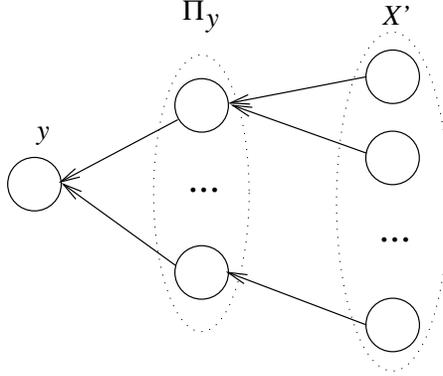


Figure 2: General structure of the synthetic BNs used in the experiments, with π_y denoting the parents of y , and with \mathcal{X}' the indirect ancestors of y .

that in each simulation the test set is the same for the two algorithms.

For the evaluation with simulated databases, we designed the experiments with the goal of assessing the capability of the scoring metrics to correctly identify the set of parents of a given variable. To this purpose, synthetic BNs were generated as follows. We considered a domain \mathcal{X} containing 8 continuous variables. One of the variables in \mathcal{X} is randomly selected to be the response variable y . A given number n_π of variables is then randomly select from $\mathcal{X} - \{y\}$ to be y 's parents. Let us denote this set of variables with π_y . All the remaining variables are randomly assigned as parents of the variables in π_y . Let us denote this set of variables with $\mathcal{X}' = \mathcal{X} - (\{y\} \cup \pi_y)$. The designation of the variables in \mathcal{X}' as parents of π_y is aimed at testing the effectiveness of the scoring metrics at identifying conditional independencies (i.e., at determining the conditional independence $y - \mathcal{X}' \mid \pi_y$). In Figure 2, we show a prototypical structure for the BNs used in the experiments.

For a given BN structure, the parameterization is in terms of finite mixtures of linear models. That is, each conditional probability $P(x_i \mid \pi_i)$ is modeled as a finite mixture of linear models as follows:

$$\begin{aligned}
 P(x_i \mid \pi_i) &= \sum_{k=1}^K \alpha_k N(\mu_k(\pi_i), \sigma_k) \\
 \mu_k(\pi_i) &= \beta_{0k} + \sum_{x_j \in \pi_i} \beta_{jk} x_j,
 \end{aligned}
 \tag{14}$$

where $N(\mu, \sigma)$ denotes a Normal distribution with mean μ and standard deviation σ . All σ_k are real numbers randomly drawn from a uniform distribution over the interval $[1, 10]$. All the regression parameters β_{jk} 's are real numbers randomly drawn from a uniform distribution over the interval $|\beta_{jk}| \in [1, K + 1]$, where K is the number of mixture components. This choice of interval is justified by the fact that we would like the resulting conditional distributions to depart significantly from a singly peaked curve, as the number of mixture components increases. Therefore, we choose to increase the magnitude of the regression parameters with the number of mixture components, in an attempt to obtain a multimodal shape for the corresponding conditional probability function. The α_k are real numbers randomly drawn from a uniform distribution over the interval $(0, 1]$, then normalized to sum to 1.

Several simulations were run for different combinations of parameter settings. In particular: i) the number n_π of parents of y in the generating synthetic network, was varied from 1 to 4; ii) the number K of mixture components used in Equation (14) was varied from 3 to 7 (Note: this is the number of linear models included in the mixtures used to *generate* the database); and iii) the number of bins used in the discretization was either 2 or 3. Furthermore, in the algorithm A2, the strategy of order selection for the mixture density network (MDN) model was either hold-out or BIC (see Section 4). Finally, we chose the maximum admissible MDN

DB	MDN order selection	P1			P2			+			-		
auto-mpg	BIC	2	2	2	2	2.5	4	0	1	2	0	0	1
	hold-out	2	2	2	1	3	3	0	1	2	0	1	1
abalone	BIC	3	4	4	0	3	4	0	1	2	1	2	4
	hold-out	3	4	4	0	1	4	0	.5	2	1	2	4

DB	MDN order selection	MSE1		MSE2		LS1		LS2		time-ratio	
auto-mpg	BIC	0.13	0.01	0.14	0.02	0.25	0.04	0.28	0.07	40.76	5.3
	hold-out	0.13	0.02	0.14	0.03	0.21	0.04	0.26	0.11	40.57	8.32
abalone	BIC	0.92	0.06	1.09	0.3	1.31	0.06	1.41	0.14	42.63	12.92
	hold-out	0.93	0.06	1.17	0.29	1.3	0.06	1.45	0.14	34.81	15.12

Figure 3: Comparison of algorithms A1 and A2 on real databases. The first table reports statistics about the structural differences of the models learned by the two algorithms. In particular, P1 and P2 are the number of parents of the response variable discovered by the two algorithms A1 and A2 respectively. The number of arcs added (+) and omitted (-) by A2 with respect to A1 are also shown. For each measure, the 3-tuple (*min*, *median*, *max*) is shown. The second table reports mean and standard deviation of the mean square error (MSE), of the log-score (LS), and of the ratio of the computational time of A2 to the computational time of A1 (time-ratio).

model order (referred to as K^{\max} in Section 4) to be 5. That is, the best model order was selected from the range $1, \dots, 5$. Finally, we ran simulations with datasets of different cardinality. In particular, we used datasets of cardinality 300, 600, and 900.

For each parameter setting, both algorithms were run five times, and in each run 20% of the database cases were randomly selected as test set, and the remaining 80% of the database cases were used for training.

We ran several simulations, whereby a simulation consists of the following steps:

- a synthetic Bayesian network B_S is generated as described above;
- a database \mathcal{D} of cases is generated from B_S by Markov simulation;
- the two algorithms A1 and A2 are applied to the database \mathcal{D} and relevant statistics on the algorithms' performance are collected.

The collected statistics for the two algorithms were then compared by means of standard statistical tests. In particular, for the continuous statistics, namely, the MSE, and the LS, we used a simple t-test, and the Welch-modified two-sample t-test for samples with unequal variance. For the discrete statistics, namely, number of arcs added and omitted, we used a median test (the Wilcoxon test).

5.2 Results

Figure 3 and Figure 4 summarize the results of the simulations with real and synthetic databases, respectively. As a general guideline, for each discrete measure (such as the number of arcs added or omitted) we report the 3-tuple (*min*, *median*, *max*). For each continuous measure (such as the log-score) we report mean and standard deviation.

With regard to the performance of the two algorithms when coupled with the two alternative MDN model selection criteria (namely, the BIC-based model selection, and the “hold-out” model selection), neither of the two criteria is significantly superior. Therefore, we report the results corresponding to the use of the BIC-based model selection only.

# of cases	MDN order selection	GS vs A1			GS vs A2			A1 vs A2											
		+			-			+			-								
300	BIC	0	0	2	0	1	4	0	0	2	0	1	3	0	0	2	0	0	2
	hold-out	0	0	2	0	1	4	0	0	2	0	1	4	0	1	3	0	0	2
600	BIC	0	0	1	0	0.5	3	0	0	3	0	1	3	0	0	3	0	0	2
	hold-out	0	0	1	0	0	3	0	1	2	0	1	3	0	1	2	0	0	3
900	BIC	0	0	1	0	0	3	0	1	3	0	0	3	0	1	3	0	0	2
	hold-out	0	0	1	0	0	3	0	0	3	0	0	4	0	1	4	0	0	2

# of cases	MDN order selection	MSE1		MSE2		LS1		LS2		time-ratio	
300	BIC	0.720	0.335	0.730	0.327	0.930	0.347	0.960	0.328	29.58	7.508
	hold-out	0.726	0.375	0.752	0.387	0.871	0.378	0.934	0.393	34.65	10.05
600	BIC	0.735	0.324	0.737	0.329	0.775	0.341	0.798	0.372	36.59	10.62
	hold-out	0.716	0.305	0.731	0.306	0.769	0.317	0.808	0.344	42.39	11.82
900	BIC	0.778	0.304	0.787	0.322	0.908	0.27	0.923	0.281	35.59	8.23
	hold-out	0.721	0.293	0.734	0.301	0.839	0.305	0.861	0.344	38.19	10.88

Figure 4: Comparison of algorithms A1 and A2 on simulated databases. In the first table, the comparison is in term of the structural differences of the discovered networks; each entry reports the 3-tuple (*min*, *median*, *max*). In the second table, the comparison is in terms of predictive accuracy; each entry reports mean and standard deviation of the quantities MSE and LS. It also reports the time ratio, given by the ratio of the computational time for A2 to the computational time for A1.

In Figure 3 we present the results of the comparison of the two learning algorithms based on the real databases. In particular we report: the number of parents P1 and P2 discovered by the two algorithms A1 and A2; the corresponding mean square error MSE and the log-score LS; and the ratio of the computational time for A2 to the computational time for A1, denoted with *time-ratio*.

In Figure 4 we present the results of the comparison of the two learning algorithms based on the simulated databases. In particular, the first table of Figure 4 reports the number of arcs added (+) and the number of arcs omitted (−) by each algorithm with respect to the gold standard Bayesian network GS (i.e., the synthetic BN used to generate the database). It also reports the number of arcs added (+) and omitted (−) by algorithm A2 with respect to algorithm A1. The second table of Figure 4 reports the measures MSE and LS for the two algorithms A1 and A2, and the time-ratio.

Notice that the statistics shown in Figure 4 are computed over different settings as stated in the previous section on experimental design.

The statistical analysis of the results reported in Figure 3 and Figure 4 shows that the difference in the prediction accuracy of the two algorithms is not statistically significant, either in terms of the mean square error, or in terms of the log-score. On the other hand, the two algorithms differ significantly when we compare the structure of the BNs they discover. In particular, with regard to the real databases, algorithm A2 tends to select a significantly larger number of parents for the response variable than algorithm A1 ($p \ll .01$). With regard to the simulated databases, algorithm A2 tends to add more extra arcs than algorithm A1 ($p \ll .01$), while the difference in the number of arcs omitted by the two algorithms is not statistically significant (remember that the number of arcs added and omitted is computed with respect to the gold standard BNs used to generate the databases). An unexpected result is the decreased prediction accuracy of both algorithms when using the dataset of 900 cases with respect to the prediction accuracy of the two algorithms when using the dataset of 600 cases. The fact that both algorithms’ performance decreases suggests that this is due to an anomaly from sampling. However the point grants further verification by testing the algorithms on larger datasets.

5.3 Discussion

The results shown in Figures 3 and 4 support the hypothesis that discretization of continuous variables does not decrease the accuracy of recovering the structure of BNs from data. They also show that using discretized continuous variables to construct a BN structure (algorithm A1) is significantly faster (by a factor ranging from about 30 to 40) than using untransformed continuous variables (algorithm A2). Also, the predictions based on A1 are at least as accurate as (and often more accurate than) the predictions based on A2.

Another important aspect differentiating the two learning methods is the relative variability of the results for algorithm A2 compared with the results for algorithm A1, especially with regard to the structure of the learned models. In Figures 3 and 4 the number of parents of the class variable discovered by algorithm A1 over multiple simulations remains basically constant (e.g., 2 parents in the database `AUTO-MPG`, 4 parents in the database `ABALONE`). This is not true for algorithm A2, where the difference between the minimum and maximum number of arcs discovered is quite high (e.g., when applied to the database `ABALONE`, A2 discovers a minimum of 0 parents and a maximum of 4 parents). These results suggest that the estimations based on the ANN-based scoring metric are not very stable, probably due to the tendency of ANN-based search to get stuck in local maxima in the search space.

6 Conclusions

In this paper, we presented a method for learning hybrid BNs, defined as BNs containing both continuous and discrete variables. The method is based on the definition of a scoring metric that makes use of artificial neural networks as probability estimators. The use of the ANN-based scoring metric allows us to search the space of BN structures without the need for discretizing the continuous variables. We compared this method to the alternative of learning the BN structure based on discretized data.

The main purpose of this work was to test whether discretization would or would not degrade the accuracy of the discovered BN structure and parameter estimation accuracy. The experimental results presented in this paper suggest that discretization of variables permits the rapid construction of relatively high fidelity Bayesian networks when compared to a much slower method that uses continuous variables. These results do not of course rule out the possibility that we can develop faster and more accurate continuous variable learning methods than the one investigated here. However, the results do lend support to discretization as a viable method for addressing the problem of learning hybrid BNs.

Acknowledgments

We thank Chris Bishop and Moises Goldszmidt for their useful comments on a preliminary version of this manuscript. This work was funded by grant IRI-9509792 from the National Science Foundation.

References

- [1] J. Binder, D. Koller, S. Russel, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 1997. To appear.
- [2] C. M. Bishop. Mixture density networks. Technical Report NCRG/4288, Neural Computing Research Group, Department of Computer Science, Aston University, Birmingham B4 7ET, U.K., February 1994.
- [3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1995.
- [4] R. Bouckært. Properties of learning algorithms for Bayesian belief networks. In *Proceedings of the 10th Conference of Uncertainty in Artificial Intelligence*, pages 102–109, San Francisco, California, 1994. Morgan Kaufmann Publishers.

- [5] J. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neuro-computing: Algorithms, Architectures and Applications*. Springer Verlag, New York, 1989.
- [6] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(3), 1996.
- [7] W. L. Buntine. Theory refinement on Bayesian networks. In *Proceedings of the 7th Conference of Uncertainty in AI*, pages 52–60, 1991.
- [8] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurasamy, editors, *Advances in Knowledge Discovery and Data Mining*. MIT Press, 1996.
- [9] D. M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: search methods and experimental results. In *Proceedings of 5th Workshop on Artificial Intelligence and Statistics*, pages 112–128, January 1995.
- [10] D. M. Chickering and D. Heckerman. Efficient approximation for the marginal likelihood of incomplete data given a Bayesian network. In *Proceedings of the 12-th Conference of Uncertainty in AI*, 1996.
- [11] G. F. Cooper. NESTOR: A computer-based medical diagnostic that integrates causal and probabilistic knowledge. Technical Report HPP-84-48, Dept. of Computer Science, Stanford University, Palo Alto, California, 1984.
- [12] G. F. Cooper and E. Herskovits. A Bayesian method for constructing Bayesian belief networks from databases. In *Proceedings of the 7th Conference of Uncertainty in Artificial Intelligence*, pages 86–94, Los Angeles, CA, 1991.
- [13] G. F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, 9:309–347, 1992.
- [14] A. Dawid. Present position and potential developments: Some personal views. Statistical theory. The prequential approach (with discussion). *Journal of Royal Statistical Society A*, 147:278–292, 1984.
- [15] A. Dawid. Prequential analysis, stochastic complexity and Bayesian inference. In J.M. Bernardo *et al*, editor, *Bayesian Statistics 4*, pages 109–125. Oxford University Press, 1992.
- [16] M. Druzdzel, L. C. van der Gaag, M. Henrion, and F. Jensen, editors. *Building probabilistic networks: where do the numbers come from?*, IJCAI-95 Workshop, Montreal, Québec, 1995.
- [17] B. Everitt and D. Hand. *Finite mixture distributions*. Chapman and Hall, 1981.
- [18] U. Fayyad and R. Uthurasamy, editors. *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, Montreal, Québec, 1995. AAAI Press.
- [19] N. Friedman and M. Goldszmidt. Discretization of continuous attributes while learning Bayesian networks. In L. Saitta, editor, *Proceedings of 13-th International Conference on Machine Learning*, pages 157–165, 1996.
- [20] D. Geiger and D. Heckerman. Learning Gaussian networks. In R. L. de Mantras and D. Poole, editors, *Proceedings of the 10th Conference of Uncertainty in AI*, San Francisco, California, 1994. Morgan Kaufmann.
- [21] D. Geiger, D. Heckerman, and C. Meek. Asymptotic model selection for directed networks with hidden variables. Technical Report MSR-TR-96-07, Microsoft Research, May 1996.
- [22] W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1996.
- [23] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.
- [24] D. Madigan, S. A. Andersson, M. D. Perlman, and C. T. Volinsky. Bayesian model averaging and model selection for Markov equivalence classes of acyclic digraphs. *Communications in Statistics – Theory and Methods*, 25, 1996.
- [25] D. Madigan, A. E. Raftery, C. T. Volinsky, and J. A. Hoeting. Bayesian model averaging. In *AAAI Workshop on Integrating Multiple Learned Models*, 1996.
- [26] C. Merz and P. Murphy. Machine learning repository. University of California, Irvine, Department of Information and Computer Science, 1996. <http://www.ics.uci.edu/mllearn/MLRepository.html>.
- [27] M. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.
- [28] S. Monti and G. F. Cooper. Learning Bayesian belief networks with neural network estimators. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference*, 1997.

- [29] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman Publishers, Inc., 1988.
- [30] A. E. Raftery. Bayesian model selection in social research (with discussion). *Sociological Methodology*, pages 111–196, 1995.
- [31] A. E. Raftery. Hypothesis testing and model selection. In Gilks *et al* [22], chapter 10, pages 163–188.
- [32] M. Richard and R. Lippman. Neural network classifiers estimate Bayesian a-posteriori probabilities. *Neural Computation*, 3:461–483, 1991.
- [33] C. Robert. Mixtures of distributions: Inference and estimation. In Gilks *et al* [22], chapter 24, pages 441–464.
- [34] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1996.
- [35] R. Shachter. Intelligent probabilistic inference. In L. K. . J. Lemmer, editor, *Uncertainty in Artificial Intelligence 1*, pages 371–382, Amsterdam, North-Holland, 1986.
- [36] D. Spiegelhalter, A. Dawid, S. Lauritzen, and R. Cowell. Bayesian analysis in expert systems. *Statistical Science*, 8(3):219–283, 1993.
- [37] B. Thiesson. Accelerated quantification of Bayesian networks with incomplete data. In Fayyad and Uthurusamy [18], pages 306–311.
- [38] D. Titterington, A. Smith, and U. Makov. *Statistical Analysis of Finite Mixture Distributions*. Wiley, New York, 1985.