, , 1–37 ()

Bayesian Network Classifiers*

NIR FRIEDMAN nir@cs.berkeley.edu
Computer Science Division, 387 Soda Hall, University of California, Berkeley, CA 94720

DAN GEIGER dang@cs.technion.ac.il
Computer Science Department, Technion, Haifa, Israel, 32000

MOISES GOLDSZMIDT moises@erg.sri.com
SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025

Editor: G. Provan, P. Langley, and P. Smyth

Abstract. Recent work in supervised learning has shown that a surprisingly simple Bayesian classifier with strong assumptions of independence among features, called *naive Bayes*, is competitive with state-of-the-art classifiers such as C4.5. This fact raises the question of whether a classifier with less restrictive assumptions can perform even better. In this paper we evaluate approaches for inducing classifiers from data, based on the theory of learning *Bayesian networks*. These networks are factored representations of probability distributions that generalize the naive Bayesian classifier and explicitly represent statements about independence. Among these approaches we single out a method we call *Tree Augmented Naive Bayes* (TAN), which outperforms naive Bayes, yet at the same time maintains the computational simplicity (no search involved) and robustness that characterize naive Bayes. We experimentally tested these approaches, using problems from the University of California at Irvine repository, and compared them to C4.5, naive Bayes, and wrapper methods for feature selection.

1. Introduction

Classification is a basic task in data analysis and pattern recognition that requires the construction of a *classifier*, that is, a function that assigns a *class* label to instances described by a set of *attributes*. The induction of classifiers from data sets of preclassified instances is a central problem in machine learning. Numerous approaches to this problem are based on various functional representations such as decision trees, decision lists, neural networks, decision graphs, and rules.

One of the most effective classifiers, in the sense that its predictive performance is competitive with state-of-the-art classifiers, is the so-called *naive Bayesian* classifier described, for example, by Duda and Hart (1973) and by Langley et al. (1992). This classifier learns from training data the conditional probability of each attribute A_i given the class label C. Classification is then done by applying Bayes rule to compute the probability of C given the particular instance of $A_1, \ldots, A_n$, and then predicting the class with the highest posterior probability. This computation is rendered feasible by making a strong independence assumption: all the attributes A_i are conditionally independent given the value of the class C. By independence we

* This paper is an extended version of Geiger (1992) and Friedman and Goldszmidt (1996a).

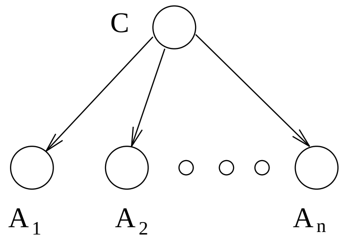


Figure 1. The structure of the naive Bayes network.

mean probabilistic independence, that is, A is independent of B given C whenever $\Pr(A|B, C) = \Pr(A|C)$ for all possible values of A, B and C, whenever $\Pr(C) > 0$.

The performance of naive Bayes is somewhat surprising, since the above assumption is clearly unrealistic. Consider, for example, a classifier for assessing the risk in loan applications: it seems counterintuitive to ignore the correlations between age, education level, and income. This example raises the following question: can we improve the performance of naive Bayesian classifiers by avoiding unwarranted (by the data) assumptions about independence?

In order to tackle this problem effectively, we need an appropriate language and efficient machinery to represent and manipulate independence assertions. Both are provided by *Bayesian networks* (Pearl, 1988). These networks are directed acyclic graphs that allow efficient and effective representation of the joint probability distribution over a set of random variables. Each vertex in the graph represents a random variable, and edges represent direct correlations between the variables. More precisely, the network encodes the following conditional independence statements: each variable is independent of its nondescendants in the graph given the state of its parents. These independencies are then exploited to reduce the number of parameters needed to characterize a probability distribution, and to efficiently compute posterior probabilities given evidence. Probabilistic parameters are encoded in a set of tables, one for each variable, in the form of local conditional distributions of a variable given its parents. Using the independence statements encoded in the network, the joint distribution is uniquely determined by these local conditional distributions.

When represented as a Bayesian network, a naive Bayesian classifier has the simple structure depicted in Figure 1. This network captures the main assumption behind the naive Bayesian classifier, namely, that every attribute (every leaf in the network) is independent from the rest of the attributes, given the state of the class variable (the root in the network). Since we have the means to represent and manipulate independence assertions, the obvious question follows: can we induce better classifiers by learning unrestricted Bayesian networks?

Learning Bayesian networks from data is a rapidly growing field of research that has seen a great deal of activity in recent years, including work by Buntine (1991, 1996), Cooper and Herskovits (1992), Friedman and Goldszmidt (1996c), Lam and

Bacchus (1994), Heckerman (1995), and Heckerman, Geiger, and Chickering (1995). This is a form of *unsupervised* learning, in the sense that the learner does not distinguish the class variable from the attribute variables in the data. The objective is to induce a network (or a set of networks) that "best describes" the probability distribution over the training data. This optimization process is implemented in practice by using heuristic search techniques to find the best candidate over the space of possible networks. The search process relies on a scoring function that assesses the merits of each candidate network.

We start by examining a straightforward application of current Bayesian networks techniques. We learn networks using the score based on the *minimum description length* (MDL) principle (Lam & Bacchus, 1994; Suzuki, 1993), and use them for classification. The results, which are analyzed in Section 3, are mixed: although the learned networks perform significantly better than naive Bayes on some data sets, they perform worse on others. We trace the reasons for these results to the definition of the MDL scoring function. Roughly speaking, the problem is that the MDL score measures the error of the learned Bayesian network over all the variables in the domain. Minimizing this error, however, does not necessarily minimize the local error in predicting the class variable given the attributes. We argue that similar problems will occur with other scoring functions in the literature.

Accordingly, we limit our attention to a class of network structures that are based on the structure of naive Bayes, requiring that the class variable be a parent of every attribute. This ensures that, in the learned network, the probability $\Pr(C|A_1, \ldots, A_n)$, the main term determining the classification, will take every attribute into account. Unlike the naive Bayesian classifier, however, our classifier allows additional edges between attributes that capture correlations among them. This extension incurs additional computational costs. While the induction of the naive Bayesian classifier requires only simple bookkeeping, the induction of Bayesian networks requires searching the space of all possible networks—that is, the space of all possible combinations of edges. To address this problem, we examine a restricted form of correlation edges. The resulting method, which we call *Tree Augmented Naive Bayes* (TAN), approximates the interactions between attributes by using a tree structure imposed on the naive Bayesian structure. As we show, this approximation is optimal, in a precise sense; moreover, we can learn TAN classifiers in polynomial time. This result extends a well-known result by Chow and Liu (1968) (see also Pearl (1988)) for learning tree-structured Bayesian networks. Finally, we also examine a generalization of these models based on the idea that correlations among attributes may vary according to the specific instance of the class variable. Thus, instead of one TAN model we have a collection of networks as the classifier. Interestingly enough, Chow and Liu already investigated classifiers of this type for recognition of handwritten characters.

After describing these methods, we report the results of an empirical evaluation comparing them with state-of-the-art machine learning methods. Our experiments show that TAN maintains the robustness and computational complexity of naive Bayes, and at the same time displays better accuracy. We compared TAN with

C4.5, naive Bayes, and *selective naive Bayes* (a wrapper approach to feature subset selection method combined with naive Bayes), on a set of problems from the University of California at Irvine (UCI) repository (see Section 4.1). These experiments show that TAN is a significant improvement over the other three approaches. We obtained similar results with a modified version of Chow's and Liu's original method, which eliminates errors due to variance in the parameters. It is interesting that this method, originally proposed three decades ago, is still competitive with state-of-the-art methods developed since then by the machine learning community.

This paper is organized as follows. In Section 2 we review Bayesian networks and how to learn them. In Section 3 we examine a straightforward application of learning Bayesian networks for classification, and show why this approach might yield classifiers that exhibit poor performance. In Section 4 we examine how to address this problem using extensions to the naive Bayesian classifier. In Section 5 we describe in detail the experimental setup and results. In Section 6 we discuss related work and alternative solutions to the problems we point out in previous sections. We conclude in Section 7. Finally, Appendix A reviews several concepts from information theory that are relevant to the contents of this paper.

2. Learning Bayesian networks

Consider a finite set $\mathbf{U} = \{X_1, \ldots, X_n\}$ of discrete random variables where each variable X_i may take on values from a finite set, denoted by $\mathit{Val}(X_i)$. We use capital letters such as X, Y, Z for variable names, and lower-case letters such as x, y, z to denote specific values taken by those variables. Sets of variables are denoted by boldface capital letters such as $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, and assignments of values to the variables in these sets are denoted by boldface lowercase letters $\mathbf{x}, \mathbf{y}, \mathbf{z}$ (we use $\mathit{Val}(\mathbf{X})$ in the obvious way). Finally, let P be a joint probability distribution over the variables in $\mathbf{U}$, and let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be subsets of $\mathbf{U}$. We say that $\mathbf{X}$ and $\mathbf{Y}$ are *conditionally independent* given $\mathbf{Z}$, if for all $\mathbf{x} \in \mathit{Val}(\mathbf{X})$, $\mathbf{y} \in \mathit{Val}(\mathbf{Y})$, $\mathbf{z} \in \mathit{Val}(\mathbf{Z})$, $P(\mathbf{x} \mid \mathbf{z}, \mathbf{y}) = P(\mathbf{x} \mid \mathbf{z})$ whenever $P(\mathbf{y}, \mathbf{z}) > 0$.

A *Bayesian network* is an annotated directed acyclic graph that encodes a joint probability distribution over a set of random variables $\mathbf{U}$. Formally, a Bayesian network for $\mathbf{U}$ is a pair $B = \langle G, \Theta \rangle$. The first component, G, is a directed acyclic graph whose vertices correspond to the random variables $X_1, \ldots, X_n$, and whose edges represent direct dependencies between the variables. The graph G encodes independence assumptions: each variable X_i is independent of its nondescendants given its parents in G. The second component of the pair, namely Θ, represents the set of parameters that quantifies the network. It contains a parameter $\theta_{x_i|\Pi_{x_i}} = P_B(x_i|\Pi_{x_i})$ for each possible value x_i of X_i, and Π_{x_i} of Π_{X_i}, where Π_{X_i} denotes the set of parents of X_i in G. A Bayesian network B defines a unique joint probability distribution over $\mathbf{U}$ given by

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_B(X_i|\Pi_{X_i}) = \prod_{i=1}^{n} \theta_{X_i|\Pi_{X_i}} . \quad (1)$$

We note that one may associate a notion of minimality with the definition of a Bayesian network, as done by Pearl (1988), yet this association is irrelevant to the material in this paper.

As an example, let $\mathbf{U}^* = \{A_1, \ldots, A_n, C\}$, where the variables $A_1, \ldots, A_n$ are the *attributes* and C is the *class* variable. Consider a graph structure where the class variable is the root, that is, $\Pi_C = \emptyset$, and each attribute has the class variable as its unique parent, namely, $\Pi_{A_i} = \{C\}$ for all $1 \leq i \leq n$. This is the structure depicted in Figure 1. For this type of graph structure, Equation 1 yields $\Pr(A_1, \ldots, A_n, C) = \Pr(C) \cdot \prod_{i=1}^{n} \Pr(A_i|C)$. From the definition of conditional probability, we get $\Pr(C|A_1, \ldots, A_n) = \alpha \cdot \Pr(C) \cdot \prod_{i=1}^{n} \Pr(A_i|C)$, where α is a normalization constant. This is in fact the definition of *naive Bayes* commonly found in the literature (Langley et al., 1992).

The problem of learning a Bayesian network can be informally stated as: Given a *training set* $D = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$ of instances of $\mathbf{U}$, find a network B that *best matches* D. The common approach to this problem is to introduce a scoring function that evaluates each network with respect to the training data, and then to search for the best network according to this function. In general, this optimization problem is intractable. Yet, for certain restricted classes of networks, there are efficient algorithms requiring polynomial time in the number of variables in the network. We indeed take advantage of these efficient algorithms in Section 4.1, where we propose a particular extension to naive Bayes.

The two main scoring functions commonly used to learn Bayesian networks are the *Bayesian scoring* function (Cooper & Herskovits, 1992; Heckerman et al., 1995), and the function based on the principle of *minimal description length* (MDL) (Lam & Bacchus, 1994; Suzuki, 1993); see also Friedman and Goldszmidt (1996c) for a more recent account of this scoring function. These scoring functions are asymptotically equivalent as the sample size increases; furthermore, they are both asymptotically correct: with probability equal to one the learned distribution converges to the underlying distribution as the number of samples increases (Heckerman, 1995; Bouckaert, 1994; Geiger et al., 1996). An in-depth discussion of the pros and cons of each scoring function is beyond the scope of this paper. Henceforth, we concentrate on the MDL scoring function.

The MDL principle (Rissanen, 1978) casts learning in terms of data compression. Roughly speaking, the goal of the learner is to find a *model* that facilitates the shortest description of the original data. The length of this description takes into account the description of the model itself and the description of the data using the model. In the context of learning Bayesian networks, the model is a network. Such a network B describes a probability distribution P_B over the instances appearing in the data. Using this distribution, we can build an encoding scheme that assigns shorter code words to more probable instances. According to the MDL principle, we should choose a network B such that the combined length of the network description and the encoded data (with respect to P_B) is minimized.

Let $B = \langle G, \Theta \rangle$ be a Bayesian network, and let $D = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$ be a training set, where each $\mathbf{u}_i$ assigns a value to all the variables in $\mathbf{U}$. The MDL scoring function of a network B given a training data set D, written $MDL(B|D)$, is given by

$$MDL(B|D) = \frac{\log N}{2}|B| - LL(B|D) \ , \tag{2}$$

where $|B|$ is the number of parameters in the network. The first term represents the length of describing the network B, in that, it counts the bits needed to encode the specific network B, where $1/2 \cdot \log N$ bits are used for each parameter in Θ. The second term is the negation of the *log likelihood* of B given D:

$$LL(B|D) = \sum_{i=1}^{N} \log(P_B(\mathbf{u}_i)) \ , \tag{3}$$

which measures how many bits are needed to describe D based on the probability distribution P_B (see Appendix A). The log likelihood also has a statistical interpretation: the higher the log likelihood, the closer B is to modeling the probability distribution in the data D. Let $\hat{P}_D(\cdot)$ be the *empirical distribution* defined by frequencies of events in D, namely, $\hat{P}_D(A) = \frac{1}{N}\sum_j 1_A(\mathbf{u}_j)$ for each event $A \subseteq Val(\mathbf{U})$, where $1_A(\mathbf{u}) = 1$ if $\mathbf{u} \in A$ and $1_A(\mathbf{u}) = 0$ if $\mathbf{u} \notin A$. Applying Equation 1 to the log likelihood and changing the order of summation yields the well-known decomposition of the log likelihood according to the structure of B:

$$LL(B|D) = N \sum_{i=1}^{N} \sum_{\substack{x_i \in Val(X_i) \\ \Pi_{x_i} \in Val(\Pi_{X_i})}} \hat{P}_D(x_i, \Pi_{x_i}) \log(\theta_{x_i|\Pi_{x_i}}) \ . \tag{4}$$

It is easy to show that this expression is maximized when

$$\theta_{x_i|\Pi_{x_i}} = \hat{P}_D(x_i|\Pi_{x_i}) \ . \tag{5}$$

Consequently, if we have two Bayesian networks, $B = \langle G, \Theta\rangle$ and $B' = \langle G, \Theta'\rangle$, that share the same structure G, and if Θ satisfies Equation 5, then $LL(B|D) \geq LL(B'|D)$. Thus, given a network structure, there is a closed form solution for the parameters that maximize the log likelihood score, namely, Equation 5. Moreover, since the first term of Equation 2 does not depend on the choice of parameters, this solution minimizes the MDL score. This is a crucial observation since it relieves us of searching in the space of Bayesian networks, and lets us search only in the smaller space of network structures, and then fill in the parameters by computing the appropriate frequencies from the data. Henceforth, unless we state otherwise, we will assume that the choice of parameters satisfies Equation 5.

The log likelihood score by itself is not suitable for learning the structure of the network, since it tends to favor *complete* graph structures in which every variable is connected to every other variable. This is highly undesirable, since such networks do not provide any useful representation of the independence assertions in the learned distributions. Moreover, these networks require an exponential number of parameters, most of which will have extremely high variance and will lead to poor predictions. Thus, the learned parameters in a maximal network will perfectly match the training data, but will have poor performance on test data. This problem, called *overfitting*, is avoided by the MDL score. The first term of the MDL score (Equation 2), regulates the complexity of networks by penalizing those that contain many parameters. Thus, the MDL score of a larger network might be worse (larger) than that of a smaller network, even though the former might match the data better. In practice, the MDL score regulates the number of parameters learned and helps avoid overfitting of the training data.

We stress that the MDL score is asymptotically correct. Given a sufficient number of independent samples, the best MDL-scoring networks will be arbitrarily close to the sampled distribution.

Regarding the search process, in this paper we will rely on a greedy strategy for the obvious computational reasons. This procedure starts with the empty network and successively applies local operations that maximally improve the score until a local minima is found. The operations applied by the search procedure include arc addition, arc deletion, and arc reversal.

3. Bayesian networks as classifiers

Using the method just described, one can induce a Bayesian network B, that encodes a distribution $P_B(A_1, \ldots, A_n, C)$, from a given training set. We can then use the resulting model so that given a set of attributes $a_1, \ldots, a_n$, the classifier based on B returns the label c that maximizes the posterior probability $P_B(c|a_1, \ldots, a_n)$. Note that, by inducing classifiers in this manner, we are addressing the main concern expressed in the introduction: we remove the bias introduced by the independence assumptions embedded in the naive Bayesian classifier.

This approach is justified by the asymptotic correctness of the Bayesian learning procedure. Given a large data set, the learned network will be a close approximation for the probability distribution governing the domain (assuming that instances are sampled independently from a fixed distribution). Although this argument provides us with a sound theoretical basis, in practice we may encounter cases where the learning process returns a network with a relatively good MDL score that performs poorly as a classifier.

To understand the possible discrepancy between good predictive accuracy and good MDL score, we must re-examine the MDL score. Recall that the log likelihood term in Equation 2 is the one that measures the quality of the learned model, and that $D = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$ denotes the training set. In a classification task, each $\mathbf{u}_i$ is a tuple of the form $\langle a_1^i, \ldots, a_n^i, c^i \rangle$ that assigns values to the attributes $A_1, \ldots, A_n$ and to the class variable C. We can rewrite the log likelihood function (Equation 3) as

$$LL(B|D) = \sum_{i=1}^{N} \log P_B(c^i|a_1^i, \ldots, a_n^i) + \sum_{i=1}^{N} \log P_B(a_1^i, \ldots, a_n^i) \ . \qquad (6)$$

The first term in this equation measures how well B estimates the probability of the class given the attributes. The second term measures how well B estimates the joint distribution of the attributes. Since the classification is determined by $P_B(C|A_1, \ldots, A_n)$, only the first term is related to the score of the network as a classifier (i.e., its predictive accuracy). Unfortunately, this term is dominated by the second term when there are many attributes; as n grows larger, the probability of each particular assignment to $A_1, \ldots, A_n$ becomes smaller, since the number of possible assignments grows exponentially in n. Thus, we expect the terms of the form $P_B(A_1, \ldots, A_n)$ to yield values closer to zero, and consequently $-\log P_B(A_1, \ldots, A_n)$ will grow larger. However, at the same time, the conditional probability of the class will remain more or less the same. This implies that a relatively large error in the conditional term in Equation 6 may not be reflected in the MDL score. Thus, using MDL (or other nonspecialized scoring functions) for learning unrestricted Bayesian networks may result in a poor classifier in cases where there are many attributes. We use the phrase "unrestricted networks" in the sense that the structure of the graph is not constrained, as in the case of a naive Bayesian classifier.

To confirm this hypothesis, we conducted an experiment comparing the classification accuracy of Bayesian networks learned using the MDL score (i.e., classifiers based on unrestricted networks) to that of the naive Bayesian classifier. We ran this experiment on 25 data sets, 23 of which were from the UCI repository (Murphy & Aha, 1995). Section 5 describes in detail the experimental setup, evaluation methods, and results. As the results in Figure 2 show, the classifier based on unrestricted networks performed significantly better than naive Bayes on six data sets, but performed significantly worse on six data sets. A quick examination of the data sets reveals that all the data sets on which unrestricted networks performed poorly contain more than 15 attributes.

A closer inspection of the networks induced on the two data sets where the unrestricted networks performed substantially worse reveals that in these networks the number of *relevant* attributes influencing the classification is rather small. While these data sets ("soybean-large" and "satimage") contain 35 and 36 attributes, respectively, the classifiers induced relied only on five attributes for the class prediction. We base our definition of relevant attributes on the notion of a *Markov blanket* of a variable X, which consists of X's parents, X's children, and the parents of X's children in a given network structure G (Pearl, 1988). This set has the property that, conditioned on X's Markov blanket, X is independent of all other variables in the network. In particular, given an assignment to all the attributes in the Markov blanket of the class variable C, the class variable is independent of the rest of the attributes. Hence, prediction using a classifier based on a Bayesian network examines only the values of attributes in the Markov blanket of C. (Note that in the naive Bayesian classifier, the Markov blanket of C includes *all* the attributes, since

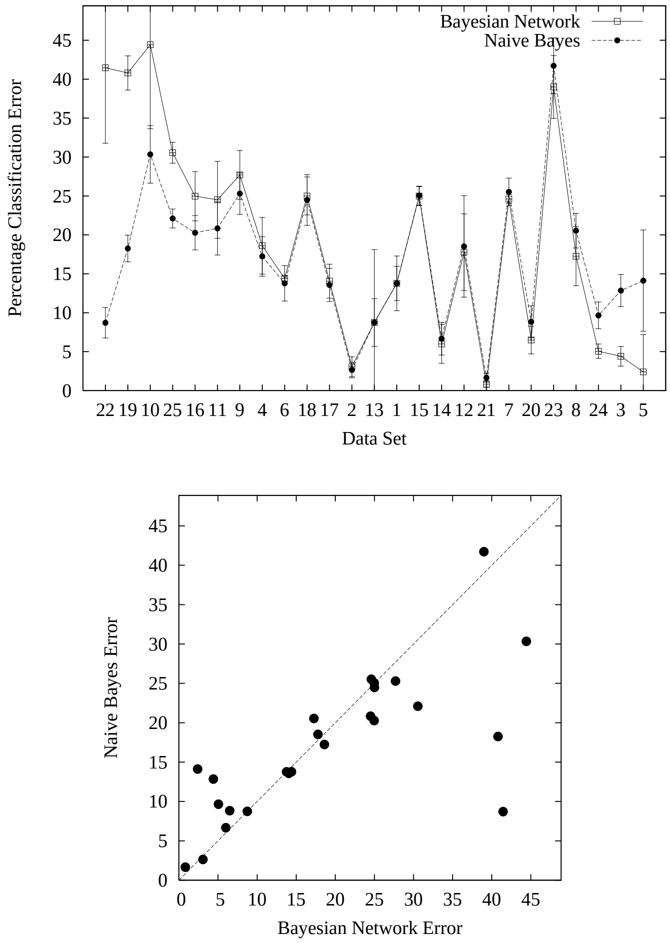


Figure 2. Error curves (top) and scatter plot (bottom) comparing unsupervised Bayesian networks (solid line, x axis) to naive Bayes (dashed line, y axis). In the error curves, the horizontal axis lists the data sets, which are sorted so that the curves cross only once, and the vertical axis measures the percentage of test instances that were misclassified (i.e., prediction errors). Thus, the smaller the value, the better the accuracy. Each data point is annotated by a 90% confidence interval. In the scatter plot, each point represents a data set, where the x coordinate of a point is the percentage of misclassifications according to unsupervised Bayesian networks and the y coordinate is the percentage of misclassifications according to naive Bayes. Thus, points above the diagonal line correspond to data sets on which unrestricted Bayesian networks perform better, and points below the diagonal line correspond to data sets on which naive Bayes performs better.

all of the attributes are children of C in the graph.) Thus, in learning the structure of the network, the learning algorithm chooses the attributes that are relevant for predicting the class. In other words, the learning procedure performs a *feature selection*. Often, this selection is useful and discards truly irrelevant attributes. However, as these two examples show, the procedure might discard attributes that are crucial for classification. The choices made by the learning procedure reflect the bias of the MDL score, which penalizes the addition of these crucial attributes to the class variable's Markov blanket. As our analysis suggests, the root of the problem is the scoring function—a network with a better score is not necessarily a better classifier.

A straightforward approach to this problem would be to specialize the scoring function (MDL in this case) to the classification task. We can do so by restricting the log likelihood to the first term of Equation 6. Formally, let the *conditional log likelihood* of a Bayesian network B given data set D be $CLL(B|D) = \sum_{i=1}^{N} \log P_B(C^i|A_1^i, \ldots, A_n^i)$. The problem associated with the application of this conditional scoring function in practice is of a computational nature. The function does not decompose over the structure of the network; that is, we do not have an analogue of Equation 4. As a consequence, setting the parameters $\theta_{x_i|\Pi_{x_i}} = \hat{P}_D(x_i|\Pi_{x_i})$ no longer maximizes the score for a fixed network structure. Thus, we would need to implement, in addition, a procedure to maximize this new function over the space of parameters. We discuss this issue further in Section 6.2. Alternative approaches are discussed in the next section.

4. Extensions to the naive Bayesian classifier

In this section we examine approaches that maintain the basic structure of a naive Bayes classifier, and thus ensure that all attributes are part of the class variable Markov blanket. These approaches, however, remove the strong assumptions of independence in naive Bayes by finding correlations among attributes that are warranted by the training data.

4.1. Augmented naive Bayesian networks as classifiers

We argued above that the performance of a Bayesian network as a classifier may improve if the learning procedure takes into account the special status of the class variable. An easy way to ensure this is to bias the structure of the network, as in the naive Bayesian classifier, such that there is an edge from the class variable to each attribute. This ensures that, in the learned network, the probability $P(C|A_1, \ldots, A_n)$ will take all attributes into account. In order to improve the performance of a classifier based on this bias, we propose to augment the naive Bayes structure with edges among the attributes, when needed, thus dispensing with its strong assumptions about independence. We call these structures *augmented naive Bayesian networks* and these edges *augmenting edges.*

In an augmented structure, an edge from A_i to A_j implies that the influence of A_i on the assessment of the class variable also depends on the value of A_j.

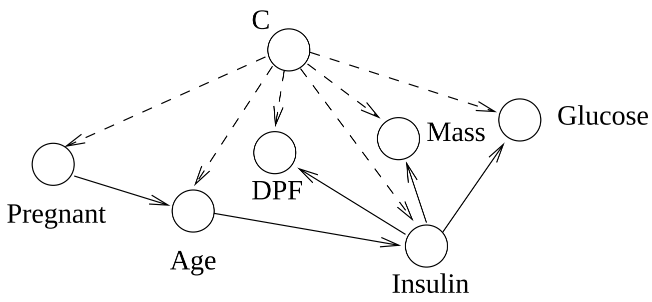


Figure 3. A TAN model learned for the data set "pima." The dashed lines are those edges required by the naive Bayesian classifier. The solid lines are correlation edges between attributes.

For example, in Figure 3, the influence of the attribute "Glucose" on the class C depends on the value of "Insulin," while in the naive Bayesian classifier the influence of each attribute on the class variable is independent of other attributes. These edges affect the classification process in that a value of "Glucose" that is typically surprising (i.e., $P(g|c)$ is low) may be unsurprising if the value of its correlated attribute, "Insulin," is also unlikely (i.e., $P(g|c,i)$ is high). In this situation, the naive Bayesian classifier will overpenalize the probability of the class variable by considering two unlikely observations, while the augmented network of Figure 3 will not.

Adding the best set of augmenting edges is an intractable problem, since it is equivalent to learning the best Bayesian network among those in which C is a root. Thus, even if we could improve the performance of a naive Bayes classifier in this way, the computational effort required may not be worthwhile. However, by imposing acceptable restrictions on the form of the allowed interactions, we can actually learn the optimal set of augmenting edges in polynomial time.

Our proposal is to learn a *tree-augmented naive Bayesian* (TAN) network in which the class variable has no parents and each attribute has as parents the class variable and at most one other attribute.[1] Thus, each attribute can have one augmenting edge pointing to it. The network in Figure 3 is in fact an TAN model. As we now show, we can take advantage of this restriction to learn a TAN model efficiently. The procedure for learning these edges is based on a well-known method reported by Chow and Liu (CL from now on) (1968), for learning tree-like Bayesian networks (see also (Pearl, 1988, pp. 387–390)). We start by reviewing CL's result.

A directed acyclic graph on $\{X_1, \ldots, X_n\}$ is a *tree* if Π_{X_i} contains exactly one parent for all X_i, except for one variable that has no parents (this variable is referred to as the *root*). A tree network can be described by identifying the parent of each variable. A function $\pi : \{1, \ldots, n\} \mapsto \{0, \ldots, n\}$ is said to *define* a tree over $X_1, \ldots, X_n$, if there is exactly one i such that $\pi(i) = 0$ (namely the root of the tree), and there is no sequence $i_1, \ldots, i_k$ such that $\pi(i_j) = i_{j+1}$ for $i \leq j < k$

and $\pi(i_k) = i_1$ (i.e., no cycles). Such a function defines a tree network where $\Pi_{X_i} = \{X_{\pi(i)}\}$ if $\pi(i) > 0$, and $\Pi_{X_i} = \emptyset$ if $\pi(i) = 0$.

Chow and Liu (1968) describe a procedure for constructing a tree Bayesian network from data. This procedure reduces the problem of constructing a maximum likelihood tree to finding a *maximal weighted spanning tree* in a graph. The problem of finding such a tree is to select a subset of arcs from a graph such that the selected arcs constitute a tree and the sum of weights attached to the selected arcs is maximized. There are well-known algorithms for solving this problem of time complexity $O(n^2 \log n)$, where n is the number of vertices in the graph (Cormen et al., 1990).

The `Construct-Tree` procedure of CL consists of four steps:

1. Compute $I_{\hat{P}_D}(X_i; X_j)$ between each pair of variables, $i \neq j$, where

$$I_P(\mathbf{X}; \mathbf{Y}) = \sum_{\mathbf{x},\mathbf{y}} P(\mathbf{x}, \mathbf{y}) \log \frac{P(\mathbf{x}, \mathbf{y})}{P(\mathbf{x})P(\mathbf{y})}$$

   is the *mutual information* function. Roughly speaking, this function measures how much information $\mathbf{Y}$ provides about $\mathbf{X}$. See Appendix A for a more detailed description of this function.

2. Build a complete undirected graph in which the vertices are the variables in $\mathbf{X}$. Annotate the weight of an edge connecting X_i to X_j by $I_{\hat{P}_D}(X_i; X_j)$.

3. Build a maximum weighted spanning tree.

4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it.

CL prove that this procedure finds the tree that maximizes the likelihood given the data D.

THEOREM 1 (Chow & Liu, 1968) *Let D be a collection of N instances of $X_1, \ldots, X_n$. The* `Construct-Tree` *procedure constructs a tree B_T that maximizes $LL(B_T|D)$ and has time complexity $O(n^2 \cdot N)$.*

This result can now be adapted to learn the maximum likelihood TAN structure. Let $A_1, \ldots, A_n$ be a set of attribute variables and C be the class variable. We say that B is a TAN model if $\Pi_C = \emptyset$ and there is a function π that defines a tree over $A_1, \ldots, A_n$ such that $\Pi_{A_i} = \{C, A_{\pi(i)}\}$ if $\pi(i) > 0$, and $\Pi_{A_i} = \{C\}$ if $\pi(i) = 0$. The optimization problem consists on finding a tree defining function π over $A_1, \ldots, A_n$ such that the log likelihood is maximized.

As we prove below, the procedure we call `Construct-TAN` solves this optimization problem. This procedure follows the general outline of CL's procedure, except that instead of using the mutual information between two attributes, it uses *conditional* *mutual information* between attributes given the class variable. This function is defined as

$$I_P(\mathbf{X};\mathbf{Y}|\mathbf{Z}) = \sum_{\mathbf{x},\mathbf{y},\mathbf{z}} P(\mathbf{x},\mathbf{y},\mathbf{z}) \log \frac{P(\mathbf{x},\mathbf{y}|\mathbf{z})}{P(\mathbf{x}|\mathbf{z})P(\mathbf{y}|\mathbf{z})} .$$

Roughly speaking, this function measures the information that $\mathbf{Y}$ provides about $\mathbf{X}$ when the value of $\mathbf{Z}$ is known. Again, Appendix A gives a more detailed description of this function.

The `Construct-TAN` procedure consists of five main steps:

1. Compute $I_{\hat{P}_D}(A_i; A_j \mid C)$ between each pair of attributes, $i \neq j$.

2. Build a complete undirected graph in which the vertices are the attributes $A_1, \ldots, A_n$. Annotate the weight of an edge connecting A_i to A_j by $I_{\hat{P}_D}(A_i; A_j \mid C)$.

3. Build a maximum weighted spanning tree.

4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it.

5. Construct a TAN model by adding a vertex labeled by C and adding an arc from C to each A_i.

THEOREM 2 *Let D be a collection of N instances of $C, A_1, \ldots, A_n$. The procedure* `Construct-TAN` *builds a TAN B_T that maximizes $LL(B_T|D)$ and has time complexity $O(n^2 \cdot N)$.*

Proof: We start with a reformulation of the log likelihood:

$$LL(B_T|D) = N \cdot \sum_{X_i} I_{\hat{P}_D}(X_i; \Pi_{X_i}) + \text{constant term}, \tag{7}$$

which we derive in Appendix A. Thus, maximizing the log likelihood is equivalent to maximizing the term

$$\sum_{X_i} I_{\hat{P}_D}(X_i; \Pi_{X_i}) .$$

We now specialize this term for TAN models. Let B_T be a TAN defined by $\pi(\cdot)$. Since C has no parents, we have $I_{\hat{P}_D}(C; \Pi_C) = 0$. Since the parents of A_i are defined by π, we set $I_{\hat{P}_D}(A_i; \Pi_{A_i}) = I_{\hat{P}_D}(A_i; A_{\pi(i)}, C)$ if $\pi(i) > 0$ and $I_{\hat{P}_D}(A_i; \Pi_{A_i}) = I_{\hat{P}_D}(A_i; C)$ if $\pi(i) = 0$. Hence, we need to maximize the term

$$\sum_{i,\pi(i)>0} I_{\hat{P}_D}(A_i; A_{\pi(i)}, C) + \sum_{i,\pi(i)=0} I_{\hat{P}_D}(A_i; C) . \tag{8}$$

We simplify this term by using the identity known as the *chain law* for mutual information (Cover & Thomas, 1991): $I_P(\mathbf{X};\mathbf{Y},\mathbf{Z}) = I_P(\mathbf{X};\mathbf{Z}) + I_P(\mathbf{X};\mathbf{Y}|\mathbf{Z})$. Hence, we can rewrite expression (8) as

$$\sum_i I_{\hat{P}_D}(A_i;C) + \sum_{i,\pi(i)>0} I_{\hat{P}_D}(A_i;A_{\pi(i)}|C)$$

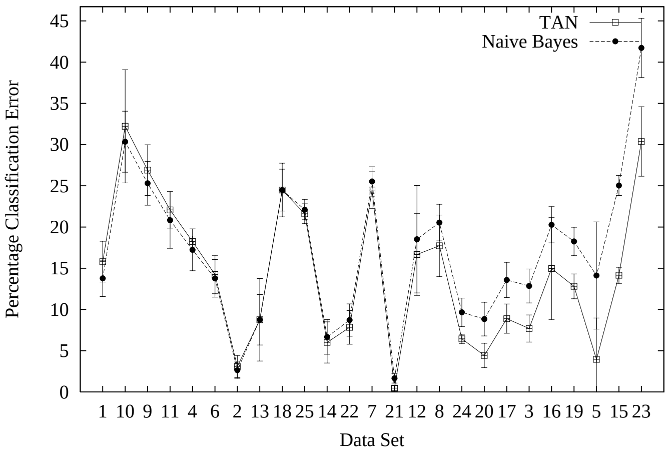
Note that the first term is not affected by the choice of $\pi(i)$. Therefore, it suffices to maximize the second term. Note also that the TAN model found by `Construct-TAN` is guaranteed to maximize this term, and thus maximizes the log likelihood.

The first step of `Construct-TAN` has complexity of $O(n^2 \cdot N)$ and the third step has complexity of $O(n^2 \log n)$. Since usually $N > \log n$, we get the stated time complexity. ■

Our initial experiments showed that the TAN model works well in that it yields good classifiers compared to naive Bayes, as shown in Tables 2 and 3). Its performance was further improved by the introduction of an additional smoothing operation. Recall that to learn the parameters of a network we estimate conditional frequencies of the form $\hat{P}_D(X|\Pi_X)$. We do this by partitioning the training data according to the possible values of Π_X and then computing the frequency of X in each partition. When some of these partitions contain very few instances, however, the estimate of the conditional probability is unreliable. This problem is not as acute in the case of a naive Bayesian classifier, since it partitions the data according to the class variable, and usually all values of the class variables are adequately represented in the training data. In TAN networks, however, for each attribute we assess the conditional probability given the class variable and another attribute. This means that the number of partitions is at least twice as large. Thus, it is not surprising to encounter unreliable estimates, especially in small data sets.

To deal with this problem, we introduce a smoothing operation on the parameters learned in TAN models that is motivated by Bayesian considerations. In Bayesian learning of a *multinomial distribution* $P(X = v_i)$ for $i = 1,\ldots,k$, we start with a *prior* probability measure over the possible settings of parameters $\Theta = \{\theta_i : i = 1,\ldots,k\}$, where $\theta_i = P(X = v_i)$, and then compute the *posterior* probability $\Pr(\Theta \mid D)$. The predicted probability for a new instance of X is the weighted average of the predictions of all possible setting of Θ, weighted by their posterior probability. Thus, $\Pr(X = v_i|D) = \int \Pr(X_i = v_i \mid \Theta) Pr(\Theta \mid D) d\Theta$. For a particular family of priors, called *Dirichlet priors*, there is a known closed-form solution for this integral. A Dirichlet prior is specified by two *hyperparameters*: Θ^0, an initial estimate of Θ, and N^0, a number that summarizes our confidence in this initial estimate. One can think of N^0 as the number of samples we have seen in our lifetime prior to making the estimate Θ^0. Given hyperparameters $\Theta^0 = \{\theta_i^0\}$ and N^0, and a data set D of length N, the prediction for $P(X = v_i)$ has the form

$$P(X = v_i|D) = \frac{N}{N + N^0}\hat{P}_D(X = v_i) + \frac{N^0}{N + N^0}\theta_i^0 \ .$$



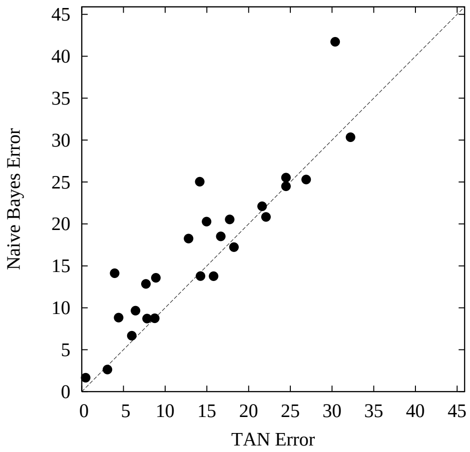


Figure 4. Error curves and scatter plot comparing smoothed TAN (solid, x axis) with naive Bayes (dashed, y axis). In the error curves, the smaller the value, the better the accuracy. In the scatter plot, points above the diagonal line correspond to data sets where smoothed TAN performs better, and points below the diagonal line correspond to data sets where naive Bayes performs better. See the caption of Figure 2 for a more detailed description.

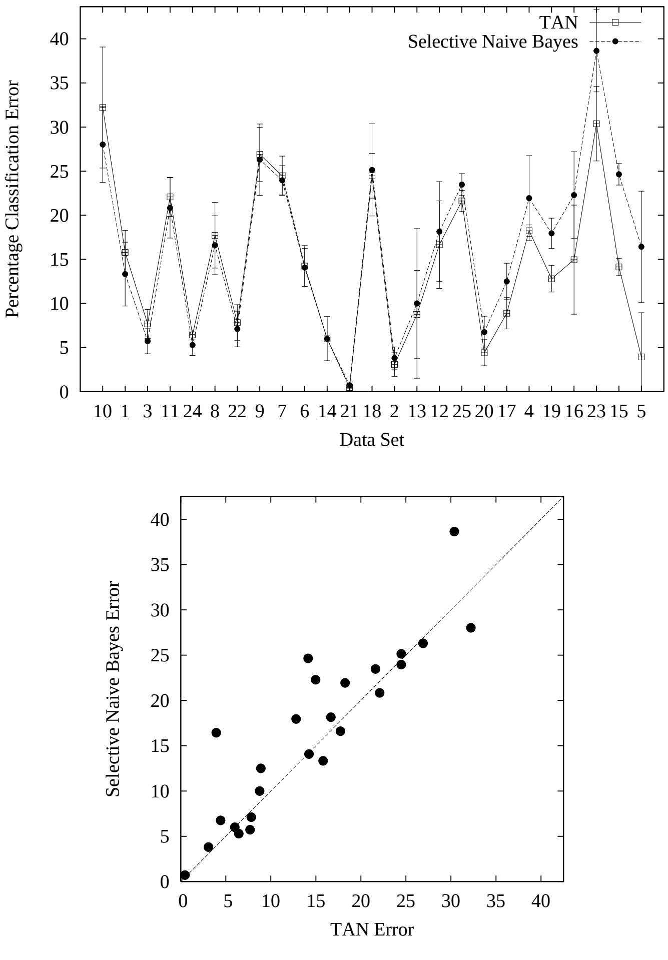


Figure 5. Error curves and scatter plot comparing smoothed TAN (solid, x axis) with selective naive Bayes (dashed, y axis). In the error curves, the smaller the value, the better the accuracy. In the scatter plot, points above the diagonal line correspond to data sets where smoothed TAN performs better, and points below the diagonal line correspond to data sets where selective naive Bayes performs better. See the caption of Figure 2 for a more detailed description.

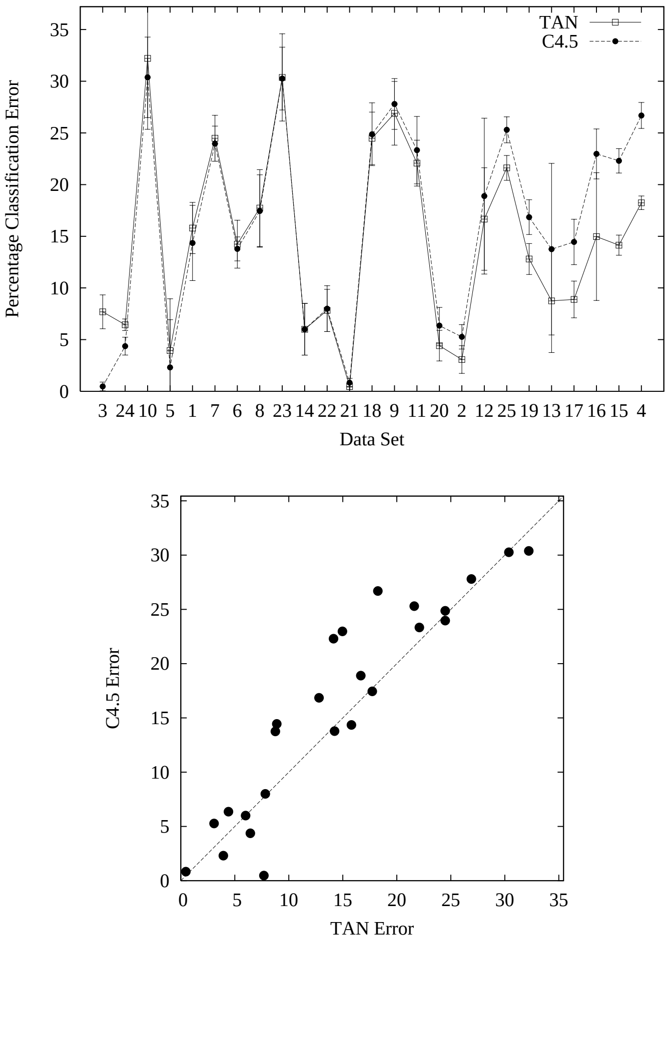


Figure 6. Error curves and scatter plot comparing smoothed TAN (solid, x axis) with C4.5 (dashed, y axis). In the error curves, the smaller the value, the better the accuracy. In the scatter plot, points above the diagonal line correspond to data sets where smoothed TAN performs better, and points below the diagonal line correspond to data set where C4.5 performs better. See the caption of Figure 2 for a more detailed description.

We refer the interested reader to DeGroot (1970). It is easy to see that this prediction biases the learned parameters in a manner that depends on the confidence in the prior and the number of new instances in the data: the more instances we have in the training data, the less bias is applied. If the number of instances N is large relative to N^0, than the bias essentially disappears. On the other hand, if the number of instances is small, then the prior dominates.

In the context of learning Bayesian networks, we can use a different Dirichlet prior for each distribution of X_i given a particular value of its parents (Heckerman, 1995). This results in choosing the parameters

$$\theta^s(x|\Pi_x) = \frac{N \cdot \hat{P}_D(\Pi_x)}{N \cdot \hat{P}_D(\Pi_x) + N^0_{x|\Pi_x}} \cdot \hat{P}_D(x|\Pi_x) + \frac{N^0_{x|\Pi_x}}{N \cdot \hat{P}_D(\Pi_x) + N^0_{x|\Pi_x}} \theta^0(x|\Pi_x) \ ,$$

where $\theta^0(x|\Pi_x)$ is the prior estimate of $P(x|\Pi_x)$ and $N^0_{x|\Pi_x}$ is the confidence associated with that prior. Note that this application of Dirichlet priors biases the estimation of the parameters depending on the number of instances in the data with particular values of X's parents. Thus, it mainly affects the estimation in those parts of the conditional probability table that are rarely seen in the training data.

To use this method, we must therefore choose the prior parameters. One reasonable choice of prior is the uniform distribution with some small N^0. Another reasonable choice uses the marginal probability of X in the data as the prior probability. This choice is based on the assumption that most conditional probabilities are close to the observed marginal. Thus, we set $\theta^0(x \mid \Pi_x) = \hat{P}_D(x)$. After initial trials we choose the value of N^0 to be 5 in all of our experiments. (More precisely, we tried the values $1, 5$, and 10 on a few data sets, and $N^0 = 5$ was slightly better than the others.) We note that this smoothing is performed after determining the structure of the TAN model. Thus, the smoothed model has the same qualitative structure as the original model but has different numerical parameters. This form of smoothing is standard practice in Bayesian statistics.[2]

In our experiments comparing the prediction error of smoothed TAN to that of unsmoothed TAN, we observed that smoothed TAN performs at least as well as TAN, and occasionally outperforms TAN significantly (e.g., see the results for "soybean-large," "segment," and "lymphography" in Table 3). Henceforth, we will assume that the version of TAN uses the smoothing operator, unless noted otherwise.

Figure 4 compares the prediction error of the TAN classifier to that of naive Bayes. As can be seen, the the TAN classifier dominates naive Bayes. This result supports our hypothesis that, by relaxing the strong independence assumptions made by naive Bayes, one can indeed learn better classifiers. We also tried a smoothed version of naive Bayes. This, however, did not lead to significant improvement over the unsmoothed naive Bayes. The only data set where there was a noticeable improvement is "lymphography," where the smoothed version had 81.73% accuracy compared to 79.72% without smoothing. Note that for this particular data set, the smoothed version of TAN has 85.03% accuracy compared to 66.87% without smoothing. The complete results for the smoothed version of naive Bayes are reported in Table 3.

Given that TAN performs better than naive Bayes and that naive Bayes is comparable to C4.5 (Quinlan, 1993), a state-of-the-art decision tree learner, we may infer that TAN should perform rather well in comparison to C4.5. To confirm this prediction, we performed experiments comparing TAN to C4.5, and also to the *selective naive Bayesian* classifier (Langley & Sage, 1994; John & Kohavi, 1997). The latter approach searches for the subset of attributes over which naive Bayes has the best performance. The results, displayed in Figures 5 and 6 and in Table 2, show that TAN is competitive with both approaches and can lead to significant improvements in many cases.

4.2. Bayesian multinets as classifiers

The TAN approach forces the relations among attributes to be the same for all the different instances of the class variable C. An immediate generalization would have different augmenting edges (tree structures in the case of TAN) for each class, and a collection of networks as the classifier.

To implement this idea, we partition the training data set by classes. Then, for each class c_i in $\mathit{Val}(C)$, we construct a Bayesian network B_i for the attribute variables $\{A_1, \ldots, A_n\}$. The resulting probability distribution $P_{B_i}(A_1, \ldots, A_n)$ approximates the joint distribution of the attributes, given a specific class, that is, $\hat{P}_D(A_1, \ldots, A_n \mid C = c_i)$. The Bayesian network for c_i is called a *local network for* c_i. The set of local networks combined with a prior on C, $P(C)$, is called a *Bayesian multinet* (Heckerman, 1991; Geiger & Heckerman, 1996). Formally, a multinet is a tuple $M = \langle P_C, B_1, \ldots, B_k \rangle$ where P_C is a distribution on C, and B_i is a Bayesian network over $A_1, \ldots, A_n$ for $1 \leq i \leq k = |\mathit{Val}(C)|$. A multinet M defines a joint distribution:

$$P_M(C, A_1, \ldots, A_n) = P_C(C) \cdot P_{B_i}(A_1, \ldots, A_n) \text{ when } C = c_i.$$

When learning a multinet, we set $P_C(C)$ to be the frequency of the class variable in the training data, that is, $\hat{P}_D(C)$, and learn the networks B_i in the manner just described. Once again, we classify by choosing the class that maximizes the posterior probability $P_M(C|A_1, \ldots, A_n)$. By partitioning the data according to the class variable, this methodology ensures that the interactions between the class variable and the attributes are taken into account. The multinet proposal is strictly a generalization of the augmented naive Bayes, in the sense that that an augmented naive Bayesian network can be easily simulated by a multinet where all the local networks have the same structure. Note that the computational complexity of finding unrestricted augmenting edges for the attributes is aggravated by the need to learn a different network for each for each value of the class variable. Thus, the search for learning the Bayesian network structure must be carried out several times, each time on a different data set.

As in the case of augmented naive Bayes, we can address this problem by constraining the class of local networks we might learn to be treelike. Indeed, the construction of a set of trees that minimizes the log likelihood score was the original method used by Chow and Liu (1968) to build classifiers for recognizing handwritten characters. They reported that, in their experiments, the error rate of this method was less than half that of naive Bayes.

We can use the algorithm in Theorem 1 separately to the attributes that correspond to each value of the class variable. This results in a multinet in which each network is a tree.

COROLLARY 1 (Chow & Liu, 1968) *Let D be a collection of N instances of C, $A_1, \ldots, A_n$. There is a procedure of time complexity $O(n^2 \cdot N)$ which constructs a multinet consisting of trees that maximizes log likelihood.*

Proof: The procedure is as follows:

1. Split D into $k = |\mathit{Val}(C)|$ partitions, $D_1, \ldots, D_k$, such that D_i contains all the instances in D where $C = c_i$.

2. Set $P_C(c_i) = \hat{P}_D(c_i)$.

3. Apply the procedure `Construct-Tree` of Theorem 1 on D_i to construct B_i.

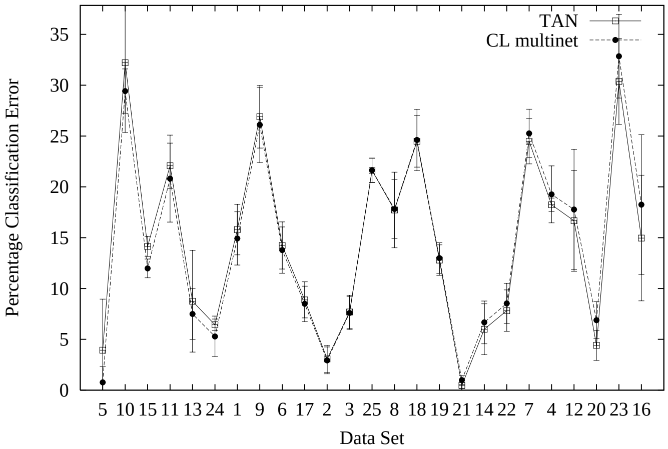
Steps 1 and 2 take linear time. Theorem 1 states that step 3 has time complexity $O(n^2|D_i|)$ for each i. Since $\sum_i |D_i| = N$, we conclude that the whole procedure has time complexity $O(n^2 N)$. ■

As with TAN models, we apply smoothing to avoid unreliable estimation of parameters. Note also that we partition the data further, and therefore run a higher risk of missing the accurate weight of some edge (in contrast to TAN). On the other hand, TAN forces the model to show the same augmenting edges for all classes. As can be expected, our experiments (see Figure 7) show that Chow and Liu (CL) multinets perform as well as TAN, and that neither approach clearly dominates.

4.3. Beyond tree-like networks

In the previous two sections we concentrated our attention on tree-like augmented naive Bayesian networks and Bayesian multinets, respectively. This restriction was motivated mainly by computational considerations: these networks can be induced in a provably effective manner. This raises the question whether we can achieve better performance at the cost of computational efficiency. One straightforward approach to this question is to search the space of all augmented naive Bayesian networks (or the larger space of Bayesian multinets) and select the one that minimizes the MDL score.

This approach presents two problems. First, we cannot examine all possible network structures; therefore we must resort to heuristic search. In this paper we



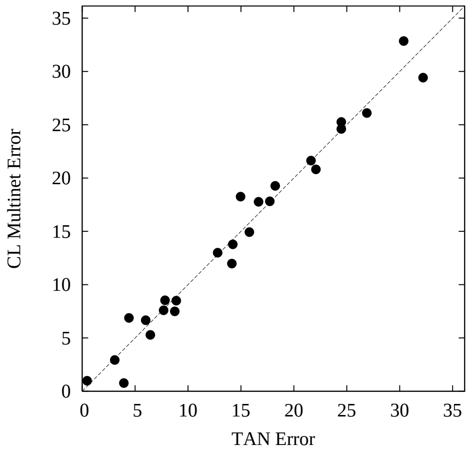


Figure 7. Error curves and scatter plot comparing smoothed TAN (solid line, x axis) with smoothed CL multinet classifier (dashed line, y axis). In the error curves, the smaller the value, the better the accuracy. In the scatter plot, points above the diagonal line corresponds to data sets where smoothed TAN performs better and points below the diagonal line corresponds to data sets where smoothed CL multinets classifier performs better. See the caption of Figure 2 for a more detailed description.

have examined a greedy search procedure. Such a procedure usually finds a good approximation to the minimal MDL scoring network. Occasionally, however, it will stop at a "poor" local minimum. To illustrate this point, we ran this procedure on a data set generated from a parity function. This concept can be captured by augmenting the naive Bayes structure with a complete subgraph. However, the greedy procedure returned the naive Bayes structure, which resulted in a poor classification rate. The greedy procedure learns this network because attributes are independent of each other given the class. As a consequence, the addition of any single edge did not improve the score, and thus, the greedy procedure terminated without adding any edges.

The second problem involves the MDL score. Recall that the MDL score penalizes larger networks. The relative size of the penalty grows larger for smaller data sets, so that the score is heavily biased for simple networks. As a result, the procedure we just described might learn too few augmenting edges. This problem is especially acute when there are many classes. In this case, the naive Bayesian structure by itself requires many parameters, and the addition of an augmenting edge involves adding at least as many parameters as the number of classes. In contrast, we note that both the TAN and CL multinet classifier learn a spanning tree over all attributes.

As shown by our experimental results, see Table 4, both unrestricted augmented naive Bayesian networks and unrestricted multinets lead to improved performance over that of the unrestricted Bayesian networks of Section 3. Moreover, on some data sets they have better accuracy than TAN and CL multinets.

5. Experimental methodology and results

We ran our experiments on the 25 data sets listed in Table 1. All of the data sets come from the UCI repository (Murphy & Aha, 1995), with the exception of "mofn-3-7-10" and "corral". These two artificial data sets were designed by John and Kohavi (1997) to evaluate methods for feature subset selection.

The accuracy of each classifier is based on the percentage of successful predictions on the test sets of each data set. We used the MLC++ system (Kohavi et al., 1994) to estimate the prediction accuracy for each classifier, as well as the variance of this accuracy. Accuracy was measured via the holdout method for the larger data sets (that is, the learning procedures were given a subset of the instances and were evaluated on the remaining instances), and via five-fold *cross validation*, using the methods described by Kohavi (1995), for the smaller ones.[3] Since we do not deal, at present, with missing data, we removed instances with missing values from the data sets. Currently, we also do not handle continuous attributes. Instead, we applied a pre-discretization step in the manner described by Dougherty et al. (1995). This pre-discretization is based on a variant of Fayyad and Irani's (1993) discretization method. These preprocessing stages were carried out by the MLC++ system. Runs with the various learning procedures were carried out on the same training sets and

Table 1. Description of data sets used in the experiments.

	Dataset	# Attributes	# Classes	# Instances Train	Test
1	australian	14	2	690	CV-5
2	breast	10	2	683	CV-5
3	chess	36	2	2130	1066
4	cleve	13	2	296	CV-5
5	corral	6	2	128	CV-5
6	crx	15	2	653	CV-5
7	diabetes	8	2	768	CV-5
8	flare	10	2	1066	CV-5
9	german	20	2	1000	CV-5
10	glass	9	7	214	CV-5
11	glass2	9	2	163	CV-5
12	heart	13	2	270	CV-5
13	hepatitis	19	2	80	CV-5
14	iris	4	3	150	CV-5
15	letter	16	26	15000	5000
16	lymphography	18	4	148	CV-5
17	mofn-3-7-10	10	2	300	1024
18	pima	8	2	768	CV-5
19	satimage	36	6	4435	2000
20	segment	19	7	1540	770
21	shuttle-small	9	7	3866	1934
22	soybean-large	35	19	562	CV-5
23	vehicle	18	4	846	CV-5
24	vote	16	2	435	CV-5
25	waveform-21	21	3	300	4700

evaluated on the same test sets. In particular, the cross-validation folds were the same for all the experiments on each data set.

Table 2 displays the accuracies of the main classification approaches we have discussed throughout the paper using the abbreviations:

NB: the naive Bayesian classifier

BN: unrestricted Bayesian networks learned with the MDL score

TAN^s: TAN networks learned according to Theorem 2, with smoothed parameters

CL^s: CL multinet classifier—Bayesian multinets learned according to Theorem 1—with smoothed parameters

C4.5: the decision-tree induction method developed by Quinlan (1993)

SNB: the *selective naive Bayesian classifier*, a wrapper-based feature selection applied to naive Bayes, using the implementation of John and Kohavi (1997)

In the previous sections we discussed these results in some detail. We now summarize the highlights. The results displayed in Table 2 show that although unrestricted Bayesian networks can often lead to significant improvement over the naive Bayesian classifier, they can also result in poor classifiers in the presence of multiple attributes. These results also show that both TAN and the CL multinet classifier are roughly equivalent in terms of accuracy, dominate the naive Bayesian classifier, and compare favorably with both C4.5 and the selective naive Bayesian classifier.

Table 3 displays the accuracies of the naive Bayesian classifier, the TAN classifier, and CL multinet classifier with and without smoothing. The columns labeled **NB**, **TAN**, and **CL** present the accuracies without smoothing, and the columns labeled $\mathbf{NB}^s$, $\mathbf{TAN}^s$, and $\mathbf{CL}^s$ describe the accuracies with smoothing. These results show that smoothing can significantly improve the accuracy both of TAN and of CL multinet classifier and does not significantly degrade the accuracy of results from other data sets. Improvement is noticed mainly in small data sets and in data sets with large numbers of classes. On the other hand, smoothing does not significantly improve the accuracy of the naive Bayesian classifier.

Finally, in Table 4 we summarize the accuracies of learning unrestricted augmented naive Bayes networks (**ANB**) and multinets (**MN**) using the MDL score. The table also contains the corresponding tree-like classifiers for comparison. These results show that learning unrestricted networks can improve the accuracy in data sets that contain strong interactions between attributes and that are large enough for the MDL score to add edges. On other data sets, the MDL score is reluctant to add edges giving structures that are similar to the naive Bayesian classifier. Consequently, in these data sets, the predictive accuracy will be poor when compared with TAN and CL multinet classifier.

6. Discussion

In this section, we review related work and expand on the issue of a conditional log likelihood scoring function. Additionally, we discuss how to extend the methods presented here to deal with complicating factors such as numeric attributes and missing values.

6.1. Related work on naive Bayes

There has been recent interest in explaining the surprisingly good performance of the naive Bayesian classifier (Domingos & Pazzani, 1996; Friedman, 1997a). The analysis provided by Friedman (1997a) is particularly illustrative, in that it focuses on characterizing how the bias and variance components of the estimation error combine to influence classification performance. For the naive Bayesian classifier, he shows that, under certain conditions, the low variance associated with this classifier can dramatically mitigate the effect of the high bias that results from the strong independence assumptions.

One goal of the work described in this paper has been to improve the performance of the naive Bayesian classifier by relaxing these independence assumptions. Indeed, our empirical results indicate that a more accurate modeling of the dependencies amongst features leads to improved classification. Previous extensions to the naive Bayesian classifier also identified the strong independence assumptions as the source

Table 2. Experimental results of the primary approaches discussed in this paper.

	Data set	NB	BN	TAN^s	CL^s	C4.5	SNB
1	australian	86.23+-1.10	86.23+-1.76	84.20+-1.24	85.07+-1.31	85.65+-1.82	**86.67+-1.81**
2	breast	**97.36+-0.50**	96.92+-0.63	96.92+-0.67	97.07+-0.66	94.73+-0.59	96.19+-0.63
3	chess	87.15+-1.03	95.59+-0.63	92.31+-0.82	92.40+-0.81	**99.53+-0.21**	94.28+-0.71
4	cleve	**82.76+-1.27**	81.39+-1.82	81.76+-0.33	80.73+-1.40	73.31+-0.63	78.06+-2.41
5	corral	85.88+-3.25	97.60+-2.40	96.06+-2.51	**99.23+-0.77**	97.69+-2.31	83.57+-3.15
6	crx	**86.22+-1.14**	85.60+-0.17	85.76+-1.16	**86.22+-1.14**	**86.22+-0.58**	85.92+-1.08
7	diabetes	74.48+-0.89	75.39+-0.29	75.52+-1.11	74.74+-1.19	**76.04+-0.85**	**76.04+-0.83**
8	flare	79.46+-1.11	82.74+-1.90	82.27+-1.86	82.18+-1.45	82.55+-1.75	**83.40+-1.67**
9	german	**74.70+-1.33**	72.30+-1.57	73.10+-1.54	73.90+-1.85	72.20+-1.23	73.70+-2.02
10	glass	69.66+-1.85	55.57+-5.39	67.78+-3.43	70.58+-1.09	69.62+-1.95	**71.98+-2.15**
11	glass2	**79.17+-1.71**	75.49+-2.47	77.92+-1.11	**79.19+-2.14**	76.67+-1.63	**79.17+-1.71**
12	heart	81.48+-3.26	82.22+-2.46	**83.33+-2.48**	82.22+-2.96	81.11+-3.77	81.85+-2.83
13	hepatitis	91.25+-1.53	91.25+-4.68	91.25+-2.50	**92.50+-1.25**	86.25+-4.15	90.00+-4.24
14	iris	93.33+-1.05	**94.00+-1.25**	**94.00+-1.25**	93.33+-1.05	**94.00+-1.25**	**94.00+-1.25**
15	letter	74.96+-0.61	75.02+-0.61	85.86+-0.49	**88.02+-0.46**	77.70+-0.59	75.36+-0.61
16	lymphography	79.72+-1.10	75.03+-1.58	**85.03+-3.09**	81.75+-3.44	77.03+-1.21	77.72+-2.46
17	mofn-3-7-10	86.43+-1.07	85.94+-1.09	91.11+-0.89	**91.50+-0.87**	85.55+-1.10	87.50+-1.03
18	pima	**75.51+-1.63**	75.00+-1.22	**75.52+-1.27**	75.39+-1.51	75.13+-1.52	74.86+-2.61
19	satimage	81.75+-0.86	59.20+-1.10	**87.20+-0.75**	87.00+-0.75	83.15+-0.84	82.05+-0.86
20	segment	91.17+-1.02	93.51+-0.89	**95.58+-0.74**	93.12+-0.91	93.64+-0.88	93.25+-0.90
21	shuttle-small	98.34+-0.29	99.17+-0.21	**99.53+-0.15**	99.02+-0.22	99.17+-0.21	99.28+-0.19
22	soybean-large	91.29+-0.98	58.54+-4.84	92.17+-1.02	91.46+-0.99	92.00+-1.11	**92.89+-1.01**
23	vehicle	58.28+-1.79	61.00+-2.02	69.63+-2.11	67.15+-2.06	**69.74+-1.52**	61.36+-2.33
24	vote	90.34+-0.86	94.94+-0.46	93.56+-0.28	94.71+-1.00	**95.63+-0.43**	94.71+-0.59
25	waveform-21	77.89+-0.61	69.45+-0.67	**78.38+-0.60**	**78.36+-0.60**	74.70+-0.63	76.53+-0.62

Table 3. Experimental results describing the effect of smoothing parameters.

	Data set	**NB**	**NB**^{s}	**TAN**	**TAN**^{s}	**CL**	**CL**^{s}
1	australian	**86.23+-1.10**	**86.23+-1.10**	81.30+-1.06	84.20+-1.24	82.17+-1.27	85.07+-1.31
2	breast	**97.36+-0.50**	**97.36+-0.55**	95.75+-1.25	96.92+-0.67	95.90+-1.13	97.07+-0.66
3	chess	87.15+-1.03	87.05+-1.03	**92.40+-0.81**	92.31+-0.82	**92.40+-0.81**	**92.40+-0.81**
4	cleve	**82.76+-1.27**	**82.76+-1.27**	79.06+-0.65	81.76+-0.33	78.04+-1.20	80.73+-1.40
5	corral	85.88+-3.25	85.88+-3.25	95.32+-2.26	96.06+-2.51	**99.23+-0.77**	**99.23+-0.77**
6	crx	**86.22+-1.14**	**86.22+-1.14**	83.77+-1.34	85.76+-1.16	83.92+-1.50	**86.22+-1.14**
7	diabetes	74.48+-0.89	74.48+-0.89	75.13+-0.98	**75.52+-1.11**	74.35+-1.43	74.74+-1.19
8	flare	79.46+-1.11	79.65+-1.23	**82.74+-1.60**	82.27+-1.86	81.90+-1.51	82.18+-1.45
9	german	**74.70+-1.33**	74.60+-1.34	72.20+-1.54	73.10+-1.54	71.80+-2.31	73.90+-1.85
10	glass	69.66+-1.85	67.78+-2.17	69.18+-2.64	67.78+-3.43	69.17+-1.29	**70.58+-1.09**
11	glass2	79.17+-1.71	**79.77+-1.50**	79.17+-1.71	77.92+-1.11	79.17+-1.71	79.19+-2.14
12	heart	81.48+-3.26	81.48+-3.26	82.96+-2.51	**83.33+-2.48**	82.96+-3.12	82.22+-2.96
13	hepatitis	91.25+-1.53	86.25+-2.34	85.00+-2.50	91.25+-2.50	86.25+-3.06	**92.50+-1.25**
14	iris	93.33+-1.05	**94.00+-1.25**	93.33+-1.05	**94.00+-1.25**	93.33+-1.05	93.33+-1.05
15	letter	74.96+-0.61	74.66+-0.62	83.44+-0.53	85.86+-0.49	84.34+-0.51	**88.02+-0.46**
16	lymphography	79.72+-1.10	81.72+-2.62	66.87+-3.37	**85.03+-3.09**	64.11+-4.77	81.75+-3.44
17	mofn-3-7-10	86.43+-1.07	86.23+-1.08	**91.70+-0.86**	91.11+-0.89	91.60+-0.87	91.50+-0.87
18	pima	**75.51+-1.63**	**75.51+-1.63**	75.13+-1.36	**75.52+-1.27**	75.39+-1.45	75.39+-1.51
19	satimage	81.75+-0.86	81.65+-0.87	77.55+-0.93	**87.20+-0.75**	77.95+-0.93	87.00+-0.75
20	segment	91.17+-1.02	90.91+-1.04	85.32+-1.28	**95.58+-0.74**	82.86+-1.36	93.12+-0.91
21	shuttle-small	98.34+-0.29	98.76+-0.25	98.86+-0.24	**99.53+-0.15**	98.86+-0.24	99.02+-0.22
22	soybean-large	91.29+-0.98	92.00+-1.32	58.17+-1.43	**92.17+-1.02**	53.38+-1.11	91.46+-0.99
23	vehicle	58.28+-1.79	58.99+-1.57	67.86+-2.92	**69.63+-2.11**	66.56+-2.18	67.15+-2.06
24	vote	90.34+-0.86	89.89+-1.60	89.20+-1.61	93.56+-0.28	89.42+-1.72	**94.71+-1.00**
25	waveform-21	77.89+-0.61	**78.68+-0.60**	75.38+-0.63	78.38+-0.60	73.85+-0.64	78.36+-0.60

Table 4. Experimental results of comparing tree-like networks with unrestricted augmented naive Bayes and multinets.

	Data set	**TAN**^{s}	**ANB**	**CL**^{s}	**MN**
1	australian	84.20+-1.24	**86.81+-0.42**	85.07+-1.31	86.52+-0.63
2	breast	96.92+-0.67	96.49+-1.09	**97.07+-0.66**	96.63+-0.95
3	chess	92.31+-0.82	94.18+-0.72	92.40+-0.81	**96.34+-0.58**
4	cleve	**81.76+-0.33**	80.08+-1.30	80.73+-1.40	**81.76+-0.33**
5	corral	96.06+-2.51	98.40+-1.60	**99.23+-0.77**	**99.20+-0.80**
6	crx	85.76+-1.16	**86.37+-0.38**	86.22+-1.14	**86.37+-0.54**
7	diabetes	75.52+-1.11	75.52+-1.06	74.74+-1.19	**76.04+-0.75**
8	flare	82.27+-1.86	**82.84+-1.76**	82.18+-1.45	82.65+-1.61
9	german	73.10+-1.54	73.20+-1.28	**73.90+-1.85**	72.20+-2.00
10	glass	67.78+-3.43	69.66+-1.85	**70.58+-1.09**	65.91+-1.63
11	glass2	77.92+-1.11	**79.17+-1.71**	**79.19+-2.14**	77.90+-1.21
12	heart	83.33+-2.48	82.59+-2.31	82.22+-2.96	**83.70+-2.58**
13	hepatitis	91.25+-2.50	88.75+-2.34	**92.50+-1.25**	90.00+-1.53
14	iris	**94.00+-1.25**	93.33+-1.05	93.33+-1.05	93.33+-1.05
15	letter	85.86+-0.49	76.60+-0.60	**88.02+-0.46**	80.10+-0.56
16	lymphography	**85.03+-3.09**	83.10+-2.19	81.75+-3.44	79.75+-0.97
17	mofn-3-7-10	91.11+-0.89	86.43+-1.07	**91.50+-0.87**	86.43+-1.07
18	pima	75.52+-1.27	74.74+-1.23	75.39+-1.51	**76.30+-1.24**
19	satimage	**87.20+-0.75**	80.50+-0.89	87.00+-0.75	77.10+-0.94
20	segment	**95.58+-0.74**	91.17+-1.02	93.12+-0.91	90.26+-1.07
21	shuttle-small	**99.53+-0.15**	98.91+-0.24	99.02+-0.22	98.97+-0.23
22	soybean-large	**92.17+-1.02**	**92.18+-1.02**	91.46+-0.99	87.01+-1.17
23	vehicle	**69.63+-2.11**	67.38+-1.38	67.15+-2.06	64.20+-2.57
24	vote	93.56+-0.28	89.66+-1.21	**94.71+-1.00**	90.11+-1.48
25	waveform-21	**78.38+-0.60**	77.75+-0.61	**78.36+-0.60**	76.85+-0.62

of classification errors, but differ in how they address this problem. These works fall into two categories.

Work in the first category, such as that of Langley and Sage (1994) and of John and Kohavi (1997), has attempted to improve prediction accuracy by rendering some of the attributes irrelevant. The rationale is as follows. As we explained in Section 4.1, if two attributes, say A_i and A_j, are correlated, then the naive Bayesian classifier may overamplify the weight of the evidence of these two attributes on the class. The proposed solution in this category is simply to ignore one of these two attributes. (Removing attributes is also useful if some attributes are irrelevant, since they only introduce noise in the classification problem.) This is a straightforward application of *feature subset selection*. The usual approach to this problem is to search for a good subset of the attributes, using an estimation scheme, such as cross validation, to repeatedly evaluate the predictive accuracy of the naive Bayesian classifier on various subsets. The resulting classifier is called the *selective naive Bayesian classifier*, following Langley and Sage (1994).

It is clear that, if two attributes are perfectly correlated, then the removal of one can only improve the performance of the naive Bayesian classifier. Problems arise, however, if two attributes are only partially correlated. In these cases the removal of an attribute may lead to the loss of useful information, and the selective naive Bayesian classifier may still retain both attributes. In addition, this wrapper-based approach is, in general, computationally expensive. Our experimental results (see Figure 6) show that the methods we examine here are usually more accurate than the selective naive Bayesian classifier as used by John and Kohavi (1997).

Work in the second category (Kononenko, 1991; Pazzani, 1995; Ezawa & Schuermann, 1995) are closer in spirit to our proposal, since they attempt to improve the predictive accuracy by removing some of the independence assumptions. The *semi-naive Bayesian classifier* (Kononenko, 1991) is a model of the form

$$P(C, A_1, \ldots, A_n) = P(C) \cdot P(\mathcal{A}_1|C) \cdots P(\mathcal{A}_k|C) \tag{9}$$

where $\mathcal{A}_1, \ldots, \mathcal{A}_k$ are pairwise disjoint groups of attributes. Such a model assumes that A_i is conditionally independent of A_j if, and only if, they are in different groups. Thus, no assumption of independence is made about attributes that are in the same group. Kononenko's method uses statistical tests of independence to partition the attributes into groups. This procedure, however, tends to select large groups, which can lead to overfitting problems. The number of parameters needed to estimate $P(\mathcal{A}_i|C)$ is $|\mathit{Val}(C)| \cdot (\prod_{A_j \in \mathcal{A}_i} |\mathit{Val}(A_j)| - 1)$, which grows exponentially with the number of attributes in the group. Thus, the parameters assessed for $P(\mathcal{A}_i|C)$ may quickly become unreliable if $\mathcal{A}_i$ contains more than two or three attributes.

Pazzani suggests that this problem can be solved by using a cross-validation scheme to evaluate the accuracy of a classifier. His procedure starts with singleton groups (i.e., $\mathcal{A}_1 = \{A_1\}, \ldots, \mathcal{A}_n = \{A_n\}$) and then combines, in a greedy manner, pairs of groups. (He also examines a procedure that performs feature subset selection after the stage of joining attributes.) This procedure does not, in general, select large groups, since these lead to poor prediction accuracy in the cross-validation test. Thus, Pazzani's procedure learns classifiers that partition the attributes in to many small groups. Since each group of attributes is considered independent of the rest given the class, these classifiers can capture only small number of correlations among the attributes.

Both Kononenko and Pazzani essentially assume that all of the attributes in each group $\mathcal{A}_i$ can be arbitrarily correlated. To understand the implications of this assumption, we use a Bayesian network representation. If we let $\mathcal{A}_i = \{A_{i_1}, \ldots, A_{i_l}\}$, then, using the chain rule, we get

$$P(\mathcal{A}_i|C) = P(A_{i_1}|C) \cdot P(A_{i_2}|A_{i,1}, C) \cdots P(A_{i_l}|A_{i,1}, \ldots, A_{i_{l-1}}, C).$$

Applying this decomposition to each of the terms in Equation 9, we get a product form from which a Bayesian network can be built. Indeed, this is an augmented naive Bayes network, in which there is a complete subgraph—that is, one to which we cannot add arcs without introducing a cycle—on the variables of each group $\mathcal{A}_i$. In contrast, in a TAN network there is a tree that spans over all attributes; thus, these models retain conditional independencies among correlated attributes. For example, consider a data set where the two attributes, A_1 and A_2, are each correlated with another attribute, A_3, but are independent of each other given A_3. These correlations are captured by the semi-naive Bayesian classifier only if all three attributes are in the same group. In contrast, a TAN classifier can place an edge from A_3 to A_1 and another to A_2. These edges capture the correlations between the attributes. Moreover, if the attributes and the class variable are boolean, then the TAN model would require 11 parameters, while the semi-naive Bayesian classifier would require 14 parameters. Thus, the representational tools of Bayesian networks let us relax the independence assumptions between attributes in a gradual and flexible manner, and study and characterize these tradeoffs with the possibility of selecting the right compromise for the application at hand.

Ezawa and Schuermann (1995) describe a method that use correlations between attributes in a different manner. First, it computes all pairwise mutual information between attributes and sorts them in descending order. Then, the method adds edges among attributes going in the computed order until it reaches some predefined threshold T. This approach presents a problem. Consider three attributes that are correlated as in the above example: A_1 and A_3 are correlated, A_2 and A_3 are correlated, but A_1 is probabilistically independent of A_2 given A_3. When this method is used, the pairwise mutual information of all combinations will appear to be high, and the algorithm will propose an edge between every pair of attributes. Nonetheless, the edge between A_1 and A_2 is superfluous, since their relation is mediated through A_3. This problem is aggravated if we consider a fourth attribute, A_4, that is strongly correlated to A_2. Then, either more superfluous edges will be added, or, if the threshold T is reached, this genuine edge will be ignored in favor of a superfluous one. Even though the TAN approach also relies on pairwise computation of the mutual information, it avoids this problem by restricting the types of interactions to the form of a tree. We reiterate, that under this restriction, the TAN approach finds an optimal tree (see Theorem 2). This example also shows why learning structures that are not trees—that is, where some attributes have more than one parent—requires us to examine higher-order interactions such as the mutual information of A_1 with A_2 given C and A_3.

Finally, another related effort that is somewhere between the categories mentioned above is reported by Singh and Provan (1995, 1996). They combine several feature subset selection strategies with an unsupervised Bayesian network learning routine. This procedure, however, can be computationally intensive (e.g., some of their strategies (Singh & Provan, 1995) involve repeated calls to a the Bayesian network learning routine).

6.2. The conditional log likelihood

Even though the use of log likelihood is warranted by an asymptotic argument, as we have seen, it may not work well when we have a limited number of samples. In Section 3 we suggested an approach based on the decomposition of Equation 6 that evaluates the predictive error of a model by restricting the log likelihood to the first term of the equation. This approach is an example of a *node monitor*, in the terminology of Spiegelhalter, Dawid, Lauritzen, and Cowell (1993). Let the *conditional log likelihood* of a Bayesian network B, given data set D, be $CLL(B|D) = \sum_{i=1}^{N} \log P_B(C^i|A_1^i, \ldots, A_n^i)$. Maximizing this term amounts to maximizing the ability to correctly predict C for each assignment to $A_1, \ldots, A_n$. Using manipulations analogous to the one described in Appendix A, it is easy to show that maximizing the conditional log likelihood is equivalent to minimizing the *conditional cross-entropy*:

$$D(\hat{P}_D(C|A_1 \ldots, A_n) \| P_B(C|A_1 \ldots, A_n)) = \sum_{a_1,\ldots,a_n} \hat{P}_D(a_1 \ldots, a_n) D(\hat{P}_D(C|a_1 \ldots, a_n) \| P_B(C|a_1 \ldots, a_n)) \quad (10)$$

This equation shows that by maximizing the conditional log likelihood we are learning the model that best approximates the conditional probability of C given the attribute values. Consequently, the model that maximizes this scoring function should yield the best classifier.

We can easily derive a conditional MDL score that is based on the conditional log likelihood. In this variant, the learning task is stated as an attempt to efficiently describe the class values for a fixed collection of attribute records. The term describing the length of the encoding of the Bayesian network model remains as before, while the second term is equal to $N \cdot CLL(B|D)$. Unfortunately, we do not have an effective way to maximize the term $CLL(B|D)$, and thus the computation of the network that minimizes the overall score becomes infeasible.

Recall that, as discussed in Section 3, once the structure of the network is fixed, the MDL score is minimized by simply substituting the frequencies in the data as the parameters of the network (see Equation 5). Once we change the score to the

$CLL(B|D)$, this is true only for a very restricted class of structures. If C is a leaf in the network—that is, if C does not have any outgoing arcs—then it is easy to prove that setting parameters according to Equation 5 maximizes $CLL(B|D)$ for a fixed network structure. However, if C has outgoing arcs, we cannot describe $P_B(C|A_1, \ldots, A_n)$ as a product of parameters in Θ, since $P_B(C|A_1, \ldots, A_n)$ also involves a normalization constant that requires us to sum over all values of C. As a consequence, $CLL(B|D)$ does not decompose, and we cannot maximize the choice of each conditional probability table independently of the others. Hence, we do not have a closed-form equation for choosing the optimal parameters for the conditional log likelihood score. This implies that, to maximize the choice of parameters for a fixed network structure, we must resort to search methods such as gradient descent over the space of parameters (e.g., using the techniques of (Binder et al., 1997)). When learning the network structure, this search must be repeated for each structure candidate, rendering the method computationally expensive. Whether we can find heuristic approaches that will allow effective learning using the conditional log likelihood remains an open question.

The difference between procedures that maximize log likelihood and ones that maximize conditional log likelihood is similar to a standard distinction made in the statistics literature. Dawid (1976) describes two paradigms for estimating $P(C, A_1, \ldots, A_n)$. These paradigms differ in how they decompose $P(C, A_1, \ldots, A_n)$. In the *sampling* paradigm, we assume that $P(C, A_1, \ldots, A_n) = P(C) \cdot P(A_1, \ldots, A_n|C)$ and assess both terms. In the *diagnostic* paradigm, we assume that $P(C, A_1, \ldots, A_n) = P(A_1, \ldots, A_n) \cdot P(C|A_1, \ldots, A_n)$ and assess only the second term, since it is the only one relevant to the classification process. In general, neither of these approaches dominates the other (Ripley, 1996).

The naive Bayesian classifier and the extensions we have evaluated belong to the sampling paradigm. Although the unrestricted Bayesian networks (described in Section 3) do not strictly belong in either paradigm, they are closer in spirit to the sampling paradigm.

6.3. Numerical attributes and missing values

Throughout this paper we have made two assumptions: that all attributes have finite numbers of values, and that the training data are *complete*, in that each instance assigns values to all the variables of interest. We now briefly discuss how to move beyond both these restrictions.

One approach to dealing with numerical attributes is to *discretize* them prior to learning a model. This is done using a discretization procedure such as the one suggested by Fayyad and Irani (1993), to partition the range of each numerical attribute. Then we can invoke our learning method treating all variables as having discrete values. As shown by Dougherty, et al. (1995), this approach is quite effective in practice. An alternative is to discretize numerical attributes during the learning process, which lets the procedure adjust the discretization of each variable so that it contains just enough partitions to capture interactions with adjacent variables in the network. Friedman and Goldszmidt (1996b) propose a principled method for performing such discretization.

Finally, there is no conceptual difficulty in representing *hybrid* Bayesian networks that contain both discrete and continuous variables. This approach involves choosing an appropriate representation for the *conditional density* of a numerical variable given its parents; for example, Heckerman and Geiger (1995) examine learning networks with Gaussian distributions. It is straightforward to combine such representations in the classes of Bayesian networks described in this paper. For example, a Gaussian variant of the naive Bayesian classifier appears in Duda and Hart (1973) and a variant based on kernel estimators appears in John and Langley (1995). We suspect that there exist analogues to Theorem 2 for such hybrid networks but we leave this issue for future work.

Regarding the problem of missing values, in theory probabilistic methods provide a principled solution. If we assume that values are *missing at random* (Rubin, 1976), then we can use the *marginal likelihood* (the probability assigned to the parts of the instance that were observed) as the basis for scoring models. If the values are not missing at random, then more careful modeling must be exercised in order to include the mechanism responsible for the missing data.

The source of difficulty in learning from incomplete data, however, is that the marginal likelihood does not decompose. That is, the score cannot be written as the sum of local terms (as in Equation 4). Moreover, to evaluate the optimal choice of parameters for a candidate network structure, we must perform nonlinear optimization using either EM (Lauritzen, 1995) or gradient descent (Binder et al., 1997).

The problem of selecting the best structure is usually intractable in the presence of missing values. Several recent efforts (Geiger et al., 1996; Chickering & Heckerman, 1996) have examined approximations to the marginal score that can be evaluated efficiently. Additionally, Friedman (1997b) has proposed a variant of EM for selecting the graph structure that can efficiently search over many candidates. The computational cost associated with all of these methods is directly related to the problem of *inference* in the learned networks. Fortunately, inference in *TAN* models can be performed efficiently. For example, Friedman's method can be efficiently applied to learning *TAN* models in the presence of missing values. We plan to examine the effectiveness of this and other methods for dealing with missing values in future work.

7. Conclusions

In this paper, we have analyzed the direct application of the MDL method to learning unrestricted Bayesian networks for classification tasks. We showed that, although the MDL method presents strong asymptotic guarantees, it does not necessarily optimize the classification accuracy of the learned networks. Our analysis suggests a class of scoring functions that may be better suited to this task. These scoring functions appear to be computationally intractable, and we therefore plan to explore effective approaches based on approximations of these scoring functions.

The main contribution of our work is the experimental evaluation of the tree-augmented naive Bayesian classifiers, TAN, and the Chow and Liu multinet classifier. It is clear that in some situations, it would be useful to model correlations among attributes that cannot be captured by a tree structure (or collections of tree structures). Such models will be preferable when there are enough training instances to robustly estimate higher-order conditional probabilities. Still, both TAN and CL multinets embody a good tradeoff between the quality of the approximation of correlations among attributes and the computational complexity in the learning stage. The learning procedures are guaranteed to find the optimal tree structure, and, as our experimental results show, they approaches perform well in practice against state-of-the-art classification methods in machine learning. We therefore propose them as worthwhile tools for the machine learning community.

Acknowledgments

The authors are grateful to Denise Draper, Ken Fertig, Joe Halpern, David Heckerman, Ron Kohavi, Pat Langley, Judea Pearl, and Lewis Stiller for comments on previous versions of this paper and useful discussions relating to this work. We also thank Ron Kohavi for technical help with the MLC++ library. Most of this work was done while Nir Friedman and Moises Goldszmidt were at the Rockwell Science Center in Palo Alto, California. Nir Friedman also acknowledges the support of an IBM graduate fellowship and NSF Grant IRI-95-03109.

Appendix A

Information-theoretic interpretation of the log likelihood

Here we review several information-theoretic notions and how they let us represent the log likelihood score. We will concentrate on the essentials and refer the interested reader to Cover and Thomas (1991) for a comprehensive treatment of these notions.

Let P be a joint probability distribution over $\mathbf{U}$. The *entropy* of $\mathbf{X}$ (given P) is defined as $H_P(\mathbf{X}) = -\sum_{\mathbf{X}} P(\mathbf{x}) \log P(\mathbf{x})$. The function $H_P(\mathbf{X})$ is the optimal number of bits needed to store the value of $\mathbf{X}$, which roughly measures the amount of information carried by $\mathbf{X}$. More precisely, suppose that $\mathbf{x}_1, \ldots, \mathbf{x}_m$ is a sequence of independent samples of $\mathbf{X}$ according to $P(\mathbf{X})$, then we cannot represent $\mathbf{x}_1, \ldots, \mathbf{x}_m$ with less than $m \cdot H_P(\mathbf{X})$ bits (assuming that m is known). With this interpretation in mind, it is easy to understand the properties of the entropy. First, the entropy is always nonnegative, since the encoding length cannot be negative. Second, the entropy is zero if and only if $\mathbf{X}$ is perfectly predictable, i.e., if one value of $\mathbf{X}$ has probability 1. In this case, we can reconstruct $\mathbf{x}_1, \ldots, \mathbf{x}_m$ without looking at the encoding. Finally, the entropy is maximal when $P(\mathbf{X})$ is totally uninformative, i.e., assigns a uniform probability to $\mathbf{X}$.

Suppose that B is a Bayesian network over $\mathbf{U}$ that was learned from D, i.e., Θ satisfies Equation 5. The entropy associated with B is simply $H_{P_B}(\mathbf{U})$. Applying Equation 1 to $\log P_B(\mathbf{u})$, moving the product out of the logarithm, and changing the order of summation, we derive the equation

$$H_{P_B}(\mathbf{U}) = -\sum_i \sum_{x_i, \Pi_{x_i}} \hat{P}_D(x_i, \Pi_{x_i}) \log(\theta_{x_i|\Pi_{x_i}}) \ ,$$

from which it immediately follows that $H_{P_B}(\mathbf{U}) = -\frac{1}{N} LL(B|D)$, using Equation 4. This equality has several consequences. First, it implies that $-LL(B|D)$ is the optimal number of bits needed to describe D, assuming that P_B is the distribution from which D is sampled. This observation justifies the use of the term $-LL(B|D)$ for measuring the representation of D in the MDL encoding scheme. Second, this equality implies that maximizing the log likelihood is equivalent to searching for a model that minimizes the entropy, as shown by Lewis (1959).

This reading suggests that by maximizing the log likelihood we are minimizing the description of D. Another way of viewing this optimization process is to use *cross entropy*, which is also known as the Kullback-Leibler divergence (Kullback & Leibler, 1951). Cross entropy is a measure of distance between two probability distributions. Formally,

$$D(P(\mathbf{X}) \| Q(\mathbf{X})) = \sum_{\mathbf{x} \in Val(\mathbf{X})} P(\mathbf{x}) \log \frac{P(\mathbf{x})}{Q(\mathbf{x})} \ . \tag{A.1}$$

One information-theoretic interpretation of cross entropy is the *average redundancy* incurred in encoding when we use a wrong probability measure. Roughly speaking, we will incur an overhead of $D(P(\mathbf{X}) \| Q(\mathbf{X}))$ per instance in the encoding of samples of $P(\mathbf{X})$ when we assume that $\mathbf{X}$ is distributed according to Q. That is, an encoding of $\mathbf{x}_1, \ldots, \mathbf{x}_m$ will be $m(H_P(\mathbf{x}) + D(P(\mathbf{X}) \| Q(\mathbf{X})))$ bits long, $mD(P(\mathbf{X}) \| Q(\mathbf{X}))$ more than the optimal. Given this interpretation of cross entropy, it is not surprising that minimizing $D(\hat{P}_D(\mathbf{U}) \| P_B(\mathbf{U}))$ is equivalent to minimizing $H_{P_B}(\mathbf{U})$, and thus is also equivalent to maximizing $LL(B|D)$.

We now turn our attention to the structure of the log likelihood term. A measure related to entropy is the *conditional entropy*, which measures the entropy of $\mathbf{X}$ when we know the value of $\mathbf{Y}$: $H_P(\mathbf{X}|\mathbf{Y}) = -\sum_{\mathbf{y}} P(\mathbf{y}) \sum_{\mathbf{x}} P(\mathbf{x}|\mathbf{y}) \log P(\mathbf{x}|\mathbf{y})$. In terms of encoding, $H_P(\mathbf{X}|\mathbf{Y})$ measures the optimal number of bits needed to encode the value $\mathbf{X}$ when the value of $\mathbf{Y}$ is given. Intuitively, knowing the value of $\mathbf{Y}$ can only be useful for encoding $\mathbf{X}$ more compactly. Indeed, $H_P(\mathbf{X}|\mathbf{Y}) \leq H_P(\mathbf{X})$. The difference between these two values, called the *mutual information* between $\mathbf{X}$ and $\mathbf{Y}$, measures how much information $\mathbf{Y}$ bears on $\mathbf{X}$. Formally, the mutual information is defined as

$$I_P(\mathbf{X}; \mathbf{Y}) = H_P(\mathbf{X}) - H_P(\mathbf{X}|\mathbf{Y}) = \sum_{\mathbf{x}, \mathbf{y}} P(\mathbf{x}, \mathbf{y}) \log \frac{P(\mathbf{x}, \mathbf{y})}{P(\mathbf{x}) P(\mathbf{y})} \ .$$

Applying these definitions to Equation 4 we immediately derive the equation

$$LL(B|D) = -N \sum_i H_{\hat{P}_D}(X_i|\Pi_{X_i}) = N \sum_i I_{\hat{P}_D}(X_i;\Pi_{X_i}) - N \sum_i H_{\hat{P}_D}(X_i) \ . \tag{A.2}$$

Several observations are in order. First, notice that $H_{\hat{P}_D}(X_i)$ is independent of the choice of B. Thus, to maximize $LL(G|D)$ we must maximize only the first term. This representation provides an appealing intuition, since it amounts to maximizing the correlation between each X_i and its parents. Second, the representation lets us easily prove that complete networks maximize the log likelihood: if B' has a superset of the arcs in B, then $\Pi_{X_i} \subseteq \Pi'_{X_i}$ for all i; since $I(\mathbf{X};\mathbf{Y}) \leq I(\mathbf{X};\mathbf{Y} \cup \mathbf{Z})$, we immediately derive $LL(B|D) \leq LL(B'|D)$.

Notes

1. TAN structures were called "Bayesian conditional trees" by Geiger (1992).
2. An alternative notion of smoothing was investigated by Cestnik (1990) in the context of learning naive Bayesian classifiers.
3. The choice of $k = 5$ in our k-fold cross validation is based on the recommendations of Kohavi (1995).

References

Binder, J., D. Koller, S. Russell, & K. Kanazawa (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning, this issue.*

Bouckaert, R. R. (1994). Properties of Bayesian network learning algorithms. In R. López de Mantarás & D. Poole (Eds.), *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence* (pp. 102–109). San Francisco, CA: Morgan Kaufmann.

Buntine, W. (1991). Theory refinement on Bayesian networks. In B. D. D'Ambrosio, P. Smets, & P. P. Bonissone (Eds.), *Proceedings of the Seventh Annual Conference on Uncertainty Artificial Intelligence* (pp. 52–60). San Francisco, CA: Morgan Kaufmann.

Buntine, W. (1996). A guide to the literature on learning probabilistic networks from data. *IEEE Trans. on Knowledge and Data Engineering, 8*, 195–210.

Cestnik, B. (1990). Estimating probabilities: a crucial task in machine learning. In L. C. Aiello (Ed.), *Proceedings of the 9th European Conference on Artificial Intelligence* (pp. 147–149). London: Pitman.

Chickering, D. M. & D. Heckerman (1996). Efficient approximations for the marginal likelihood of incomplete data given a Bayesian network. In E. Horvits & F. Jensen (Eds.), *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence* (pp. 158–168). San Francisco, CA: Morgan Kaufmann.

Chow, C. K. & C. N. Liu (1968). Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Info. Theory, 14*, 462–467.

Cooper, G. F. & E. Herskovits (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning, 9*, 309–347.

Cormen, T. H., C. E. Leiserson, & R. L. Rivest (1990). *Introduction to Algorithms.* Cambridge, MA: MIT Press.

Cover, T. M. & J. A. Thomas (1991). *Elements of Information Theory.* New York: John Wiley & Sons.

Dawid, A. P. (1976). Properties of diagnostic data distributions. *Biometrics*, *32*, 647–658.

DeGroot, M. H. (1970). *Optimal Statistical Decisions.* New York: McGraw-Hill.

Domingos, P. & M. Pazzani (1996). Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In L. Saitta (Ed.), *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 105–112). San Francisco, CA: Morgan Kaufmann.

Dougherty, J., R. Kohavi, & M. Sahami (1995). Supervised and unsupervised discretization of continuous features. In A. Prieditis & S. Russell (Eds.), *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 194–202). San Francisco, CA: Morgan Kaufmann.

Duda, R. O. & P. E. Hart (1973). *Pattern Classification and Scene Analysis.* New York: John Wiley & Sons.

Ezawa, K. J. & T. Schuermann (1995). Fraud/uncollectable debt detection using a Bayesian network based learning system: A rare binary outcome with mixed data structures. In P. Besnard & S. Hanks (Eds.), *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (pp. 157–166). San Francisco, CA: Morgan Kaufmann.

Fayyad, U. M. & K. B. Irani (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (pp. 1022–1027). San Francisco, CA: Morgan Kaufmann.

Friedman, J. (1997a). On bias, variance, 0/1 - loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, *1*, 55–77.

Friedman, N. (1997b). Learning belief networks in the presence of missing values and hidden variables. In D. Fisher (Ed.), *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 125–133). San Francisco, CA: Morgan Kaufmann.

Friedman, N. & M. Goldszmidt (1996a). Building classifiers using Bayesian networks. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 1277–1284). Menlo Park, CA: AAAI Press.

Friedman, N. & M. Goldszmidt (1996b). Discretization of continuous attributes while learning Bayesian networks. In L. Saitta (Ed.), *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 157–165). San Francisco, CA: Morgan Kaufmann.

Friedman, N. & M. Goldszmidt (1996c). Learning Bayesian networks with local structure. In E. Horvits & F. Jensen (Eds.), *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence* (pp. 252-262). San Francisco, CA: Morgan Kaufmann.

Geiger, D. (1992). An entropy-based learning algorithm of Bayesian conditional trees. In D. Dubois, M. P. Wellman, B. D. D'Ambrosio, & P. Smets (Eds.), *Proceedings of the Eighth Annual Conference on Uncertainty Artificial Intelligence* (pp. 92–97). San Francisco, CA: Morgan Kaufmann.

Geiger, D. & D. Heckerman (1996). Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, *82*, 45–74.

Geiger, D., D. Heckerman, & C. Meek (1996). Asymptotic model selection for directed graphs with hidden variables. In E. Horvits & F. Jensen (Eds.), *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence* (pp. 283-290). San Francisco, CA: Morgan Kaufmann.

Heckerman, D. (1991). *Probabilistic Similarity Networks.* Cambridge, MA: MIT Press.

Heckerman, D. (1995). A tutorial on learning Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research.

Heckerman, D. & D. Geiger (1995). Learning Bayesian networks: a unification for discrete and Gaussian domains. In P. Besnard & S. Hanks (Eds.), *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (pp. 274–284). San Francisco, CA: Morgan Kaufmann.

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

John, G. & R. Kohavi (1997). Wrappers for feature subset selection. *Artificial Intelligence.* Accepted for publication. A preliminary version appears in *Proceedings of the Eleventh International Conference on Machine Learning*, 1994, pp. 121–129, under the title "Irrelevant features and the subset selection problem".

John, G. H. & P. Langley (1995). Estimating continuous distributions in Bayesian classifiers. In P. Besnard & S. Hanks (Eds.), *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (pp. 338–345). San Francisco, CA: Morgan Kaufmann.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1137–1143). San Francisco, CA: Morgan Kaufmann.

Kohavi, R., G. John, R. Long, D. Manley, & K. Pfleger (1994). MLC++: A machine learning library in C++. In *Proc. Sixth International Conference on Tools with Artificial Intelligence* (pp. 740–743). IEEE Computer Society Press.

Kononenko, I. (1991). Semi-naive Bayesian classifier. In Y. Kodratoff (Ed.), *Proc. Sixth European Working Session on Learning* (pp. 206–219). Berlin: Springer-Verlag.

Kullback, S. & R. A. Leibler (1951). On information and sufficiency. *Annals of Mathematical Statistics*, *22*, 76–86.

Lam, W. & F. Bacchus (1994). Learning Bayesian belief networks. An approach based on the MDL principle. *Computational Intelligence*, *10*, 269–293.

Langley, P., W. Iba, & K. Thompson (1992). An analysis of Bayesian classifiers. In *Proceedings, Tenth National Conference on Artificial Intelligence* (pp. 223–228). Menlo Park, CA: AAAI Press.

Langley, P. & S. Sage (1994). Induction of selective Bayesian classifiers. In R. López de Mantarás & D. Poole (Eds.), *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence* (pp. 399–406). San Francisco, CA: Morgan Kaufmann.

Lauritzen, S. L. (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, *19*, 191–201.

Lewis, P. M. (1959). Approximating probability distributions to reduce storage requirements. *Information and Control*, *2*, 214–225.

Murphy, P. M. & D. W. Aha (1995). UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Pazzani, M. J. (1995). Searching for dependencies in Bayesian classifiers. In D. Fisher & H. Lenz (Eds.), *Proceedings of the fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems.* San Francisco, CA: Morgan Kaufmann.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* San Francisco, CA: Morgan Kaufmann.

Ripley, B. D. (1996). *Pattern recognition and neural networks.* Cambridge: Cambridge University Press.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, *14*, 465–471.

Rubin, D. R. (1976). Inference and missing data. *Biometrica*, *63*, 581–592.

Singh, M. & G. M. Provan (1995). A comparison of induction algorithms for selective and non-selective Bayesian classifiers. In A. Prieditis & S. Russell (Eds.), *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 497–505). San Francisco, CA: Morgan Kaufmann.

Singh, M. & G. M. Provan (1996). Efficient learning of selective Bayesian network classifiers. In L. Saitta (Ed.), *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 453–461). San Francisco, CA: Morgan Kaufmann.

Spiegelhalter, D. J., A. P. Dawid, S. L. Lauritzen, & R. G. Cowell (1993). Bayesian analysis in expert systems. *Statistical Science*, *8*, 219–283.

Suzuki, J. (1993). A construction of Bayesian networks from databases based on an MDL scheme. In D. Heckerman & A. Mamdani (Eds.), *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence* (pp. 266–273). San Francisco, CA: Morgan Kaufmann.