Building Classifiers using Bayesian Networks

Nir Friedman
Stanford University
Dept. of Computer Science
Gates Building 1A
Stanford, CA 94305-9010
nir@cs.stanford.edu

Moises Goldszmidt*
Rockwell Science Center
444 High St., Suite 400
Palo Alto, CA 94301
moises@rpal.rockwell.com

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 examine and evaluate approaches for inducing classifiers from data, based on recent results in the theory of learning *Bayesian networks*. Bayesian networks are factored representations of probability distributions that generalize the naive Bayes 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 which are characteristic of naive Bayes. We experimentally tested these approaches using benchmark problems from the U. C. Irvine repository, and compared them against C4.5, naive Bayes, and wrapper-based feature selection methods.

1 Introduction

A somewhat simplified statement of the problem of supervised learning is as follows. Given a *training set* of labeled instances of the form $\langle a_1, \ldots, a_n \rangle, c$, construct a *classifier* f capable of predicting the value of c, given instances $\langle a'_1, \ldots, a'_n \rangle$ as input. The variables $A_1, \ldots, A_n$ are called *features* or *attributes*, and the variable C is usually referred to as the *class variable* or *label*. This is a basic problem in many applications of machine learning, and there are numerous approaches to solve it based on various functional representations such as decision trees, decision lists, neural networks, decision-graphs, rules, and many others. One of the most effective classifiers, in the sense that its predictive performance is competitive with state of the art classifiers such as C4.5 (Quinlan 1993), is the so-called *naive Bayesian* classifier (or simply *naive Bayes*) (Langley, Iba, & Thompson 1992). This classifier learns the conditional probability of each attribute A_i given the label C in the training data. Classification is then done by applying Bayes rule to compute the probability of C given the particular instantiation of $A_1, \ldots, A_n$. This computation is rendered feasible by making a strong independence assumption: all the attributes A_i are conditionally independent given the value of the label C.[1] The performance of naive Bayes is somewhat surprising given that this is clearly an unrealistic assumption. Consider for example a classifier for assessing the risk in loan applications. It would be erroneous to ignore the correlations between age, education level, and income.

This fact naturally begs the question of whether we can improve the performance of Bayesian classifiers by avoiding unrealistic assumptions about independence. In order to effectively tackle this problem we need an appropriate language and effective machinery to represent and manipulate independences. *Bayesian networks* (Pearl 1988) provide both. Bayesian networks are directed acyclic graphs that allow for efficient and effective representation of the joint probability distributions 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 statements about each random variable: each variable is independent of its non-descendants in the graph given the state of its parents. Other independences follow from these ones. These can be efficiently read from the network structure by means of a simple graph-theoretic criteria. Independences 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. Given the independences encoded in the network, the joint distribution can be reconstructed by simply multiplying these tables.

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 Bayes 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). Now that we have the means to represent and

*Current address: SRI International, 333 Ravenswood Way, Menlo Park, CA 94025, moises@erg.sri.com

[1] By independence we 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.

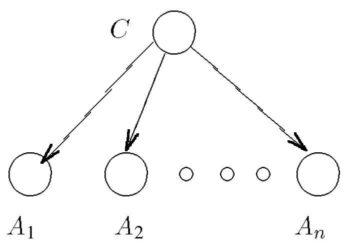

Figure 1: The structure of the naive Bayes network.

manipulate independences, the obvious question follows: can we learn an unrestricted Bayesian network from the data that when used as a classifier maximizes the prediction rate?

Learning Bayesian networks from data is a rapidly growing field of research that has seen a great deal of activity in recent years, see for example (Heckerman 1995; Heckerman, Geiger, & Chickering 1995; Lam & Bacchus 1994). This is a form *unsupervised* learning in the sense that the learner is not guided by a set of informative examples. 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 using heuristic search techniques to find the best candidate over the space of possible networks. The search process relies on a scoring metric that asses the merits of each candidate network.

We start by examining a straightforward application of current Bayesian networks techniques. We learn networks using the *MDL* metric (Lam & Bacchus 1994) 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 datasets, they perform worse on others. We trace the reasons for these results to the definition of the MDL scoring metric. Roughly speaking, the problem is that the MDL scoring metric 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 metrics in the literature.

In light of these results 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 $\Pr(C|A_1, \ldots, A_n)$ will take every attribute into account. Unlike the naive Bayes structure, however, we allow additional edges between attributes. These additional edges capture correlations among the attributes. Note that the process of adding these edges may involve an heuristic search on a subnetwork. However, there is a restricted sub-class of these structures, for which we can find the best candidate in polynomial time. This result, shown by Geiger (1992), is a consequence of a well-known result by Chow and Liu (1968) (see also (Pearl 1988)). This approach, which we call *Tree Augmented Naive Bayes* (*TAN*), approximates the interactions between attributes using a tree-structure imposed on the naive Bayes structure. We note that while this method has been proposed in the literature, it has never been rigorously tested in practice. We show that *TAN* maintains the robustness and computational complexity (the search process is bounded) of naive Bayes, and at the same time displays better performance in our experiments. We compare this approach with C4.5, naive Bayes, and *selective naive Bayes*, a wrapper-based feature subset selection method combined with naive Bayes, on a set of benchmark problems from the U. C. Irvine repository (see Section 4). This experiments show that *TAN* leads to significant improvement over all of these three approaches.

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 domain. We use capital letters, such as X, Y, Z, for variable names and lowercase letters x, y, z to denote specific values taken by those variables. The set of values X can attain is denoted as $\mathit{Val}(X)$, the cardinality of this set is denoted as $||X|| = |\mathit{Val}(X)|$. Sets of variables are denoted by boldface capital letters $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, and assignments of values to the variables in these sets will be denoted by boldface lowercase letters $\mathbf{x}, \mathbf{y}, \mathbf{z}$ (we use $\mathit{Val}(\mathbf{X})$ and $||\mathbf{X}||$ 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 $\mathbf{U}$. $\mathbf{X}$ 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 of a domain composed of a set of random variables. Formally, a Bayesian network for $\mathbf{U}$ is the pair $B = \langle G, \Theta \rangle$. G is a directed acyclic graph whose nodes correspond to the random variables $X_1, \ldots, X_n$, and whose edges represent direct dependencies between the variables. The graph structure G encodes the following set of independence assumptions: each node X_i is independent of its non-descendants given its parents in G (Pearl 1988).[2] The second component of the pair, namely Θ, represents the set of parameters that quantifies the network. It contains a parameter $\theta_{x_i|\Pi_{x_i}} = P(x_i|\Pi_{x_i})$ for each possible value x_i of X_i, and Π_{x_i} of Π_{X_i} (the set of parents of X_i in G). B defines a unique joint probability distribution over $\mathbf{U}$ given by:

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

Example 2.1: 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. In the *naive Bayes* structure the class variable

[2]Formally there is a notion of minimality associated with this definition, but we will ignore it in this paper (see (Pearl 1988)).

is the root, i.e., $\Pi_C = \emptyset$, and the only parent for each attribute is the class variable, i.e., $\Pi_{A_i} = \{C\}$, for all $1 \le i \le n$. Using (1) we have that $\Pr(A_1, \ldots, A_n, C) = \Pr(C) \cdot \prod_{i=1}^{n} \Pr(A_i|C)$. From the definition of conditional probability we get that $\Pr(C|A_1, \ldots, A_n) = \alpha \cdot \Pr(C) \cdot \prod_{i=1}^{n} \Pr(A_i|C)$, where α is a normalization constant. This is the definition of naive Bayes commonly found in the literature (Langley, Iba, & Thompson 1992). ∎

The problem of learning a Bayesian network can be stated as follows. Given a *training set* $D = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$ of instances of **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. 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 will indeed take advantage of these efficient algorithms in Section 4 where we propose a particular extension to naive Bayes. We start by examining the components of the scoring function that we will use throughout the paper.

Let $B = \langle G, \Theta \rangle$ be a Bayesian network, and let $D = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$ (where each $\mathbf{u}_i$ assigns values to all the variables in **U**) be a training set. The *log-likelihood* of B given D is defined as

$$LL(B|D) = \sum_{i=1}^{N} \log(\Pr_B(\mathbf{u}_i)). \quad (2)$$

This term measures the likelihood that the data D was generated from the model B (namely the candidate Bayesian network) when we assume that the instances were independently sampled. The higher this value is, the closer B is to modeling the probability distribution in the data D. Let $\hat{P}_D(\cdot)$ be the measure defined by frequencies of events in D. Using (1) we can decompose the the log-likelihood according to the structure of the network. After some algebraic manipulations we can easily derive:

$$LL(B|D) = N \sum_i \sum_{x_i, \Pi_{x_i}} \hat{P}_D(x_i, \Pi_{x_i}) \log(\theta_{x_i|\Pi_{x_i}}) \quad (3)$$

Now assume that the structure of the network is fixed. Standard arguments show that $LL(B|D)$ is maximized when $\theta_{x_i|\Pi_{x_i}} = \hat{P}_D(x_i|\Pi_{x_i})$.

Lemma 2.2: *Let $B = \langle G, \Theta \rangle$ and $B' = \langle G, \Theta' \rangle$ such that $\theta'_{x_i|\Pi_{x_i}} = \hat{P}_D(x_i|\Pi_{x_i})$. Then $LL(B'|D) \ge LL(B|D)$.*

Thus, we have a closed form solution for the parameters that maximize the log-likelihood for a given network structure. This is crucial since instead of searching in the space of Bayesian networks, we only need to search in the smaller space of network structures, and then fill in the parameters by computing the appropriate frequencies from the data.

The log-likelihood score, while very simple, 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 independences in the learned distributions. Moreover, the number of parameters in the complete model is exponential. Most of these parameters will have extremely high variance and will lead to poor predictions. This phenomena is called *overfitting*, since the learned parameters match the training data, but have poor performance on test data.

The two main scoring functions commonly used to learn Bayesian networks complement the log-likelihood score with additional terms to address this problem. These are the *Bayesian scoring* function (Heckerman, Geiger, & Chickering 1995), and the one based on *minimal description length* (MDL) (Lam & Bacchus 1994). In this paper we concentrate on MDL deferring the discussion on the Bayesian scoring function for the full paper.[3]

The motivation underlying the MDL method is to find a compact encoding of the training set D. We do not reproduce the derivation of the the MDL scoring function here, but merely state it. The interested reader should consult (Friedman & Goldszmidt 1996; Lam & Bacchus 1994). The MDL score of a network B given D, written $MDL(B|D)$ is

$$MDL(B|D) = \frac{1}{2} \log N |B| - LL(B|D) \quad (4)$$

where $|B|$ is the number of parameters in the network. The first term simply counts how many bits we need to encode the specific network B, where we store $1/2 \cdot \log N$ bits for each parameter in Θ. The second term measures how many bits are needed for the encoded representation of D. Minimizing the MDL score involves tradeoffs between these two factors. 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. Note that the first term does not depend on the actual parameters in B, but only on the graph structure. Thus, for a fixed the network structure, we minimize the MDL score by maximizing the LL score using Lemma 2.2.

It is important to note that learning based on the MDL score is asymptotically correct: with probability 1 the learned distribution converges to the underlying distribution as the number of samples increases (Heckerman 1995).

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 and until a local minima is found. The operations applied by the search procedure are: arc addition, arc deletion and arc reversal. In the full paper we describe results using other search methods (although the methods we examined so far did not lead to significant improvements.)

3 Bayesian Networks as Classifiers

[3]There are some well-known connections between these two proposals (Heckerman 1995).

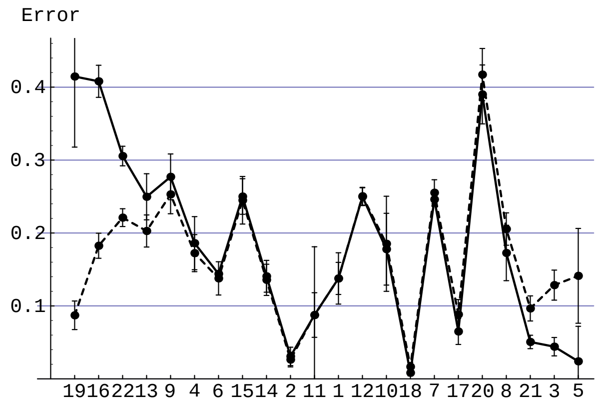

Figure 2: Error curves comparing unsupervised Bayesian networks (solid line) to naive Bayes (dashed line). The horizontal axis lists the datasets, which are sorted so that the curves cross only once. The vertical axis measures fraction of test instances that were misclassified (i.e., prediction errors). Thus, the smaller the value, the better the accuracy. Each data point is annotated by a 90% confidence interval.

Using the methods just described we can induce a Bayesian network from the data and then use the resulting model as a classifier. The learned network represents an approximation to the probability distribution governing the domain (given the assumption that the instances are independently sampled form a single distribution). Given enough samples, this will be a close approximation Thus, we can use this network to compute the probability of C given the values of the attributes. The predicted class c, given a set of attributes $a_1, \ldots, a_n$ is simply the class that attains the maximum posterior $P_B(c|a_1, \ldots, a_n)$, where P_B is the probability distribution represented by the Bayesian network B.

It is important to note that this procedure is *unsupervised* in the sense that the learning procedure does not distinguish the class variable from other attributes. Thus, we do not inform the procedure that the evaluation of the learned network will be done by measuring the predictive accuracy with respect to the class variable.

From the outset there is an obvious problem. Learning unrestricted Bayesian networks is an intractable problem. Even though in practice we resort to greedy heuristic search, this procedure is often expensive. In particular, it is more expensive than learning the naive Bayesian classifier which can be done in linear time.

Still, we may be willing to invest the extra effort required in learning a (unrestricted) Bayesian network if the prediction accuracy of the resulting classifier outperforms that of the naive Bayesian classifier. As our first experiment shows, Figure 2, this is not always the case. In this experiment we compared the predictive accuracy of classification using Bayesian networks learned in an unsupervised fashion, versus that of the naive Bayesian classifier. We run this experiment on 22 datasets, 20 of which are from the U. C. Irvine repository (Murphy & Aha 1995). Appendix A describes in detail the experimental setup, evaluation methods, and results (Table 1).

As can be seen from Figure 2 the classifier based on unsupervised networks performed significantly better than naive Bayes on 6 datasets, and performed significantly worse on 6 datasets. A quick examination of the datasets revels that all the datasets where unsupervised networks performed poorly contain more than 15 attributes.

It is interesting to examine the two datasets where the unsupervised networks performed worst (compared to naive Bayes): "soybean-large" (with 35 attributes) and "satimage" (with 36 attributes). For both these datasets, the size of the class' *Markov blanket* in the networks is rather small—less than 5 attributes. The relevance of this is that prediction using a Bayesian network examines only the values of attributes in the class variable's Markov blanket.[4] The fact that for both these datasets the Markov blanket of the class variable is so small indicates that for the MDL metric, the "cost" (in terms of additional parameters) of enlarging the Markov blanket is not worth the tradeoff in overall accuracy. Moreover, we note that in all of these experiments the networks found by the unsupervised learning routine had better MDL score than the naive Bayes network. This suggest that the root of the problem is the scoring metric—a network with a better score is not necessarily a better classifier.

To understand this problem in detail we re-examine the MDL score. Recall that the likelihood term in (4) is the one that measures the quality of the learned model. Also recall that $D = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$ is 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. Using the chain rule we can rewrite the log-likelihood function (2) as:

$$
\begin{aligned}
LL(B|D) \quad = \quad & \sum_{i=1}^{N} \log P_B(c^i | a_1^i, \ldots, a_n^i) + \\
& \sum_{i=1}^{N} \log P_B(a_1^i, \ldots, a_n^i) \qquad (5)
\end{aligned}
$$

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 prediction 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 smaller values which in turn will increase the value of the log function. However,

[4] More precisely, for a fixed network structure the Markov blanket of a variable X (Pearl 1988) consists of X's parents, X's children, and parents of X's children in G. This set has the property that conditioned on X's Markov blanket, X is independent of all other variables in the network.

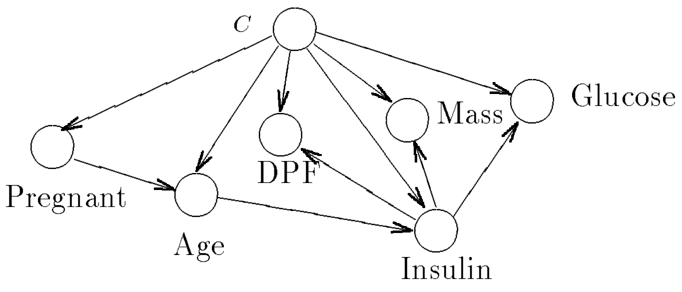

Figure 3: A TAN model learned for the dataset "pima".

at the same time, the conditional probability of the class will remain more of less the same. This implies that a relatively big error in the first term will not reflect in the MDL score. Thus, as indicated by our experimental results, using a non-specialized scoring metric for learning an unrestricted Bayesian network may result in a poor classifier when there are many attributes.[5]

A straightforward approach to dealing with this problem would be to specialize the scoring metric (MDL in this case) for the classification task. We can easily do so by restricting the log-likelihood to the first term of (5). Formally, let the *conditional log-likelihood* of a Bayesian network B given dataset D be $CLL(B|D) = \sum_{i=1}^{N} \log P_B(C^i|A_1^i, \ldots, A_n^i)$. A similar modification to Bayesian scoring metric in (Heckerman, Geiger, & Chickering 1995) is equally easy to define. The problem with applying these conditional scoring metrics in practice is that they do not decompose over the structure of the network, i.e., we do not have an analogue of (3). As a consequence it is no longer true that setting the parameters $\theta_{x_i|\Pi_{x_i}} = \hat{P}_D(x_i|\Pi_{x_i})$ maximizes the score for a fixed network structure.[6] We do not know, at this stage, whether there is a computational effective procedure to find the parameter values that maximize this type of conditional score. In fact, as reported by Geiger (1992), previous attempts to defining such conditional scores resulted in unrealistic and sometimes contradictory assumptions.

4 Learning Restricted Networks for Classifiers

In light of this discussion we examine a different approach. We limit our attention to a class of network structures that are based on the naive Bayes structure. As in naive Bayes, we require that the class variable be a parent of every attribute. This ensures that, in the learned network, the probability $P(C|A_1, \ldots, A_n)$ will take every attribute into account, rather than a shallow set of neighboring nodes.

[5]In the full paper we show that the same problem occur in the Bayesian scoring metric.

[6]We remark that decomposition still holds for a restricted class of structures—essentially these where C does not have any children. However, these structures are usually not useful for classification.

In order to improve the performance of a classifier based on naive Bayes we propose to augment the naive Bayes structure with "correlation" edges among the attributes. We call these structures *augmented naive Bayes*.

In an the augmented structure an edge from A_i to A_j implies that the two attributes are no longer independent given the class variable. In this case the influence of A_i on the assessment of class variable also depends on the value of A_j. Thus, in Figure 3, the influence of the attribute "Glucose" on the class C depends on the value of "Insulin", while in naive Bayes the influence of each on the class variable is independent of other's. Thus, a value of "Glucose" that is 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 over-penalize the probability of the class variable (by considering two unlikely observations), while the network in Figure 3 will not.

Adding the best augment naive Bayes structure is an intractable problem. To see this, note this essentially involves learning a network structure over the attribute set. However, by imposing restrictions on the form of the allowed interactions, we can learn the correlations quite efficiently.

A *tree-augmented naive Bayes* (*TAN*) model, is a Bayesian network where $\Pi_C = \emptyset$, and Π_{A_i} contains C and at most one other attribute. Thus, each attribute can have one correlation edge pointing to it. As we now show, we can exploit this restriction on the number of correlation edges to learn *TAN* models efficiently. This class of models was previously proposed by Geiger (1992), using a well-known method by Chow and Liu (1968), for learning tree-like Bayesian networks (see also (Pearl 1988, pp. 387–390)).[7]

We start by reviewing Chow and Liu's result on learning trees. A directed acyclic graph 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*). Chow and Liu show that there is a simple procedure that constructs the maximal log-probability tree. Let n be the number of random variables and N be the number of training instances. Then

Theorem 4.1: (Chow & Lui 1968) *There is a procedure of time complexity $O(n^2 \cdot N)$, that constructs the tree structure B_T that maximizes $LL(B_T|D)$.*

The procedure of Chow and Liu can be summarized as follows.

1. Compute the *mutual information* $I(X_i; X_j) = \sum_{x_i,x_j} \hat{P}_D(x_i, x_j) \log \frac{\hat{P}_D(x_i,x_j)}{\hat{P}_D(x_i)\hat{P}_D(x_j)}$ between each pair of variables, $i \neq j$.
2. Build a complete undirected graph in which the vertices are the variables in **U**. Annotate the weight of an edge connecting X_i to X_j by $I(X_i; X_j)$.
3. Build a *maximum weighted spanning tree* of this graph (Cormen, Leiserson, & Rivest 1990).

[7]These structures are called "Bayesian conditional trees" by Geiger.

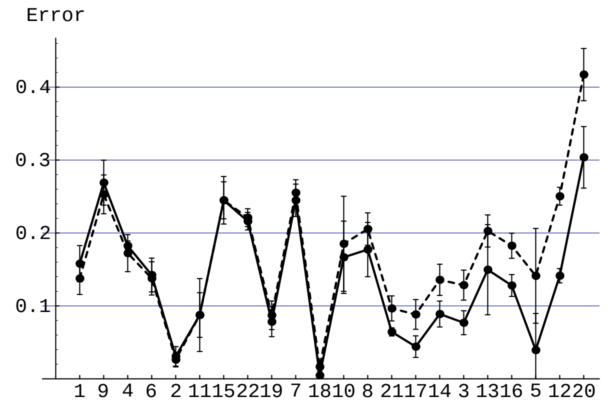

Figure 4: Error curves comparing smoothed *TAN* (solid line) to naive Bayes (dashed line).

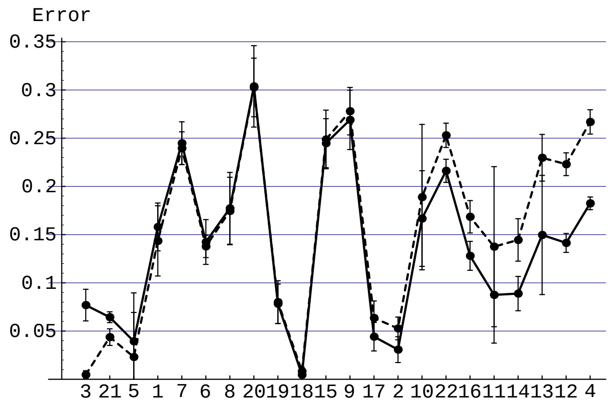

Figure 5: Error curves comparing smoothed *TAN* (solid line) to C4.5 (dashed line).

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. (The choice of root variable does not change the log-likelihood of the network.)

The first step has complexity of $O(n^2 \cdot N)$ and the third step has complexity of $O(n^2 \log n)$. Since we usually have that $N > \log n$, we get the resulting complexity.

This result can be adapted to learn the maximum likelihood *TAN* structure.

Theorem 4.2: (Geiger 1992) *There is a procedure of time complexity $O(n^2 \cdot N)$ that constructs the TAN structure B_T that maximize LL($B_T|D$).*

The procedure is very similar to the procedure described above when applied to the attributes $A_1, \ldots, A_n$. The only change is in the choice of weights. Instead of taking $I(A_i; A_j)$, we take the *conditional mutual information* given C, $I(A_i; A_j|C) = \sum_{a_i,a_j,c} \hat{P}_D(a_i, a_j, c) \log \frac{\hat{P}_D(a_i,a_j|c)}{\hat{P}_D(a_i|c)\hat{P}_D(a_j|c)}$. Roughly speaking, this measures the gain in log-likelihood of adding A_i as a parent of A_j when C is already a parent.

There is one more consideration. To learn the parameters in the network we estimate conditional frequencies of the form $\hat{P}_D(X|\Pi_X)$. This is done by partitioning the training data according to the possible values of Π_X and then computing the frequency of X in each partition. A problem surfaces when some of these partitions contain very few instances. In these small partitions our estimate of the conditional probability is unreliable. This problem is not serious for the naive Bayes 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* models however, for each attribute we asses 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 datasets.

To deal with this problem we introduce a smoothing operation on the parameters learned in *TAN* models. This operation takes every estimated parameter $\theta_{x|\Pi_x}$ and biases it toward the observed marginal frequency of X, $\hat{P}_D(x)$. Formally, we let the new parameter $\theta^s(x|\Pi_x) = \alpha \cdot \hat{P}_D(x|\Pi_x) + (1-\alpha)\hat{P}_D(x)$, taking $\alpha = \frac{N \cdot \hat{P}_D(\Pi_x)}{N \cdot \hat{P}_D(\Pi_x) + s}$, where s is the *smoothing parameter*, which is usually quite small (in all the experiments we choose $s = 5$).[8] It is easy to see that this operation biases the learned parameters in a manner that depends on our confidence level as expressed by s: the more instances we have in the partition from which we compute the parameter, the less bias is applied. If the number of instances that with a particular parents' value assignment is significant, than the bias essentially disappears. On the other hand, if the number of instances with the a particular parents' value assignment is small, then the bias dominates. We note that this operation is performed after the structure of the *TAN* model are determined. Thus, the smoothed model has exactly the same qualitative structure as the original model, but with different numerical parameters. 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 significantly outperforms *TAN* (see for example the results for "soybean-large", "segment", and "lymphography" in Table 1). From now on 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 performance of the *TAN* classifier dominates that of naive Bayes.[9] This result supports our hypothesis that by relaxing the strong independence assumptions made by naive Bayes we can indeed learn better classifiers.

[8] In statistical terms, we are essentially applying a *Dirichlet prior* on $\theta_{X|\Pi_X}$ with mean expected value $\hat{P}_D(X)$ and equivalent sample size s. We note that this use of Dirichlet priors is related to the class of Dirichlet priors described in (Heckerman, Geiger, & Chickering 1995).

[9] In our experiments we also tried smoothed version of naive Bayes. This did not lead to significant improvement over the unsmoothed naive Bayes.

Finally, we also compared *TAN* to C4.5 (Quinlan 1993), a state of the art decision-tree learning system, and to the *selective naive Bayesian* classifier (Langley & Sage 1994; John, Kohavi, & Pfleger 1995). The later approach searches for the subset of attributes over which naive Bayes has the best performance. The results displayed in Figure 5 and Table 1, show that *TAN* is quite competitive with both approaches and can lead to big improvements in many cases.

5 Concluding Remarks

This paper makes two important contributions. The first one is the analysis of unsupervised learning of Bayesian networks for classification tasks. We show that the scoring metrics used in learning unsupervised Bayesian networks do not necessarily optimize the performance of the learned networks in classification. Our analysis suggests a possible class of scoring metrics that are suited for this task. These metrics appear to be computationally intractable. We plan to explore effective approaches to learning with approximations of these scores. The second contribution is the experimental validation of tree augmented naive Bayesian classifiers, *TAN*. This approach was introduced by Geiger (1992), yet was not extensively tested and as a consequence has received little recognition in the machine learning community. This classification method has attractive computational properties, while at the same time, as our experimental results show, it performs competitively with reported state of the art classifiers.

In spite of these advantages, it is clear that in some situations, it would be useful to model correlations among attributes that cannot be captured by a tree structure. This will be significant when there is a sufficient number of training instances to robustly estimate higher-order conditional probabilities. Thus, it is interesting to examine the problem of learning (unrestricted) augmented naive Bayes networks. In an initial experiment we attempted to learn such networks using the MDL score, where we restricted the search procedure to examine only networks that contained the naive Bayes backbone. The results were somewhat disappointing, since the MDL score was reluctant to add more than a few correlation arcs to the naive Bayes backbone. This is, again, a consequence of the fact that the scoring metric is not geared for classification. An alternative approach might use a cross-validation scheme to evaluate each candidate while searching for the best correlation edges. Such a procedure, however, is computationally expensive.

We are certainly not the first to try and improve naive Bayes by adding correlations among attributes. For example, Pazzani (1995) suggests a procedure that replaces, in a greedy manner, pairs of attributes A_i, A_j, with a new attribute that represents the cross-product of A_i and A_j. This processes ensures that paired attributes influence the class variable in a correlated manner. It is easy to see that the resulting classifier is equivalent to an augmented naive Bayes network where the attributes in each "cluster" are fully interconnected. Note that including many attributes in a single cluster may result in overfitting problems. On the other hand, attributes in different clusters remain (conditionally) independent of each other. This shortcoming does not occur in *TAN* classifiers. Another example is the work by Provan and Singh (1995), in which a wrapper-based feature subset selection is applied to an unsupervised Bayesian network learning routine. This procedure is computationally intensive (it involves repeated calls to a Bayesian network learning procedure) and the reported results indicate only a slight improvement over the selective naive Bayesian classifier.

The attractiveness of the tree-augmented naive Bayesian classifier is that it embodies a good tradeoff between the quality of the approximation of correlations among attributes, and the computational complexity in the learning stage. Moreover, the learning procedure is guaranteed to find the optimal *TAN* structure. As our experimental results show this procedure performs well in practice. Therefore we propose *TAN* as a worthwhile tool for the machine learning community.

Acknowledgements

The authors are grateful to Denise Draper, Ken Fertig, Dan Geiger, Joe Halpern, Ronny Kohavi, Pat Langley and Judea Pearl for comments on a previous draft of this paper and useful discussions relating to this work. We thank Ronny Kohavi for technical help with the MLC++ library. Parts of this work were done while the first author was at Rockwell Science Center. The first author was also supported in part by an IBM Graduate fellowship and NSF Grant IRI-95-03109.

A Experimental Methodology and Results

We run our experiments on the 22 datasets listed in Table 1. All of the datasets are from the U. C. Irvine repository (Murphy & Aha 1995), with the exception of "mofn-3-7-10" and "corral". These two artificial datasets were used for the evaluation of feature subset selection methods by (John, Kohavi, & Pfleger 1995). All these datasets are accessible at the MLC++ ftp site.

The accuracy of each classifier is based on the percentage of successful predictions on the test sets of each dataset. We estimate the prediction accuracy for each classifier as well as the variance of this accuracy using the MLC++ system (Kohavi *et al.* 1994). Accuracy was evaluated using the holdout method for the larger datasets, and using 5-fold *cross validation* (using the methods described in (Kohavi 1995)) for the smaller ones. Since we do not deal, at the current time, with missing data we had removed instances with missing values from the datasets. Currently we also do not handle continuous attributes. Instead, in each invocation of the learning routine, the dataset was pre-discretized using a variant of the method of (Fayyad & Irani 1993) using only the training data, in the manner described in (Dougherty, Kohavi, & Sahami 1995). These preprocessing stages where carried out by the MLC++ system. We note that experiments with the various learning procedures were carried out on exactly the same training sets and evaluated on the same test sets. In particular, the cross-validation folds where the same for all the experiments on each dataset.

Table 1: Experimental results

	Dataset	# Attributes	# Instances		Accuracy					
			Train	Test	**NBC**	**Unsup**	**TAN**	**TAN^s**	**C4.5**	**SNBC**
1	australian	14	690	CV-5	86.23+-1.10	86.23+-1.76	81.30+-1.06	84.20+-1.24	85.65+-1.82	**86.67+-1.81**
2	breast	10	683	CV-5	**97.36+-0.50**	96.92+-0.63	95.75+-1.25	96.92+-0.67	94.73+-0.59	96.19+-0.63
3	chess	36	2130	1066	87.15+-1.03	95.59+-0.63	92.40+-0.81	92.31+-0.82	**99.53+-0.21**	94.28+-0.71
4	cleve	13	296	CV-5	**82.76+-1.27**	81.39+-1.82	79.06+-0.65	81.76+-0.33	73.31+-0.63	78.06+-2.41
5	corral	6	128	CV-5	85.88+-3.25	97.60+-2.40	95.32+-2.26	96.06+-2.51	**97.69+-2.31**	83.57+-3.15
6	crx	15	653	CV-5	**86.22+-1.14**	85.60+-0.17	83.77+-1.34	85.76+-1.16	**86.22+-0.58**	85.92+-1.08
7	diabetes	8	768	CV-5	74.48+-0.89	75.39+-0.29	75.13+-0.98	75.52+-1.11	**76.04+-0.85**	**76.04+-0.83**
8	flare	10	1066	CV-5	79.46+-1.11	82.74+-1.90	82.74+-1.60	82.27+-1.86	82.55+-1.75	**83.40+-1.67**
9	german	20	1000	CV-5	**74.70+-1.33**	72.30+-1.57	72.20+-1.54	73.10+-1.54	72.20+-1.23	73.70+-2.02
10	heart	13	270	CV-5	81.48+-3.26	82.22+-2.46	82.96+-2.51	**83.33+-2.48**	81.11+-3.77	81.85+-2.83
11	hepatitis	19	80	CV-5	**91.25+-1.53**	**91.25+-4.68**	85.00+-2.50	**91.25+-2.50**	86.25+-4.15	90.00+-4.24
12	letter	16	15000	5000	74.96+-0.61	75.02+-0.61	83.44+-0.53	**85.86+-0.49**	77.70+-0.59	75.36+-0.61
13	lymphography	18	148	CV-5	79.72+-1.10	75.03+-1.58	66.87+-3.37	**85.03+-3.09**	77.03+-1.21	77.72+-2.46
14	mofn-3-7-10	10	300	1024	86.43+-1.07	85.94+-1.09	**91.70+-0.86**	91.11+-0.89	85.55+-1.10	87.50+-1.03
15	pima	8	768	CV-5	**75.51+-1.63**	75.00+-1.22	75.13+-1.36	**75.52+-1.27**	75.13+-1.52	74.86+-2.61
16	satimage	36	4435	2000	81.75+-0.86	59.20+-1.10	77.55+-0.93	**87.20+-0.75**	83.15+-0.84	82.05+-0.86
17	segment	19	1540	770	91.17+-1.02	93.51+-0.89	85.32+-1.28	**95.58+-0.74**	93.64+-0.88	93.25+-0.90
18	shuttle-small	9	3866	1934	98.34+-0.29	99.17+-0.21	98.86+-0.24	**99.53+-0.15**	99.17+-0.21	99.28+-0.19
19	soybean-large	35	562	CV-5	91.29+-0.98	58.54+-4.84	58.17+-1.43	92.17+-1.02	92.00+-1.11	**92.89+-1.01**
20	vehicle	18	846	CV-5	58.28+-1.79	61.00+-2.02	67.86+-2.92	69.63+-2.11	**69.74+-1.52**	61.36+-2.33
21	vote	16	435	CV-5	90.34+-0.86	94.94+-0.46	89.20+-1.61	93.56+-0.28	**95.63+-0.43**	94.71+-0.59
22	waveform-21	21	300	4700	77.89+-0.61	69.45+-0.67	75.38+-0.63	**78.38+-0.60**	74.70+-0.63	76.53+-0.62

Finally, in Table 1 we summarize the accuracies of the six learning procedures we discussed in this paper: **NBC**–the naive Bayesian classifier; **Unsup**–unsupervised Bayesian networks learned using the MDL score; **TAN**—*TAN* networks learned according to Theorem 4.2; **TAN^s**—smoothed *TAN* networks; **C4.5**–the decision-tree classifier of (Quinlan 1993); **SNBC**—the selective naive Bayesian classifier, a wrapper-based feature selection applied to naive Bayes, using the implementation of (John, Kohavi, & Pfleger 1995).

References

Chow, C. K., and Lui, C. N. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Info. Theory* 14:462–467.

Cormen, T. H.; Leiserson, C. E.; and Rivest, R. L. 1990. *Introduction to Algorithms*. MIT Press.

Dougherty, J.; Kohavi, R.; and Sahami, M. 1995. Supervised and unsupervised discretization of continuous features. In *ML '95*.

Fayyad, U. M., and Irani, K. B. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI '93*, 1022–1027.

Friedman, N., and Goldszmidt, M. 1996. Discretization of continuous attributes while learning Bayesian networks. In *ML '96*.

Geiger, D. 1992. An entropy-based learning algorithm of Bayesian conditional trees. In *UAI '92*. 92–97.

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

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

John, G.; Kohavi, R.; and Pfleger, K. 1995. Irrelevant features and the subset selection problem. In *ML '94*. 121–129.

Kohavi, R.; John, G.; Long, R.; Manley, D.; and Pfleger, K. 1994. MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence*. 740–743.

Kohavi, R. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI '95*. 1137–1143.

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

Langley, P., and Sage, S. 1994. Induction of selective Bayesian classifiers. In *UAI '94*. 399–406.

Langley, P.; Iba, W.; and Thompson, K. 1992. An analysis of bayesian classifiers. In *AAAI '90*. 223–228.

Murphy, P. M., and Aha, D. W. 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 *Proc. of the 5'th Int. Workshop on Artificial Intelligence and Statistics*.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Singh, M., and Provan, G. M. 1995. A comparison of induction algorithms for selective and non-selective bayesian classifiers. In *ML '95*.