Bayesian Network Classification with Continuous Attributes: Getting the Best of Both Discretization and Parametric Fitting

Nir Friedman*
Computer Science Division
University of California
Berkeley, CA 94920
nir@cs.berkeley.ed

Moises Goldszmidt
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
moises@erg.sri.com

Thomas J. Lee
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
tomlee@erg.sri.com

Abstract

In a recent paper, Friedman, Geiger, and Goldszmidt [8] introduced a classifier based on Bayesian networks, called *Tree Augmented Naive Bayes* (TAN), that outperforms naive Bayes and performs competitively with C4.5 and other state-of-the-art methods. This classifier has several advantages including robustness and polynomial computational complexity. One limitation of the TAN classifier is that it applies only to discrete attributes, and thus, continuous attributes must be prediscretized. In this paper, we extend TAN to deal with continuous attributes directly via parametric (e.g., Gaussians) and semiparametric (e.g., mixture of Gaussians) conditional probabilities. The result is a classifier that can represent and combine both discrete and continuous attributes. In addition, we propose a new method that takes advantage of the modeling language of Bayesian networks in order to represent attributes both in discrete and continuous form simultaneously, and use both versions in the classification. This automates the process of deciding which form of the attribute is most relevant to the classification task. It also avoids the commitment to either a discretized or a (semi)parametric form, since different attributes may correlate better with one version or the other. Our empirical results show that this latter method usually achieves classification performance that is as good as or better than either the purely discrete or the purely continuous TAN models.

1 INTRODUCTION

The effective handling of continuous attributes is a central problem in machine learning and pattern recognition. Almost every real-world domain, including medicine, industrial control, and finance, involves continuous attributes. Moreover, these attributes usually have rich interdependencies with other discrete attributes. Many approaches in machine learning deal with continuous attributes by discretizing them. In statistics and pattern recognition, on the other hand, the typical approach is to use a parametric family of distributions (e.g. Gaussians) to model the data.

Each of these strategies has its advantages and disadvantages. By using a specific parametric family, we are making strong assumptions about the nature of the data. If these assumptions are warranted, then the induced model can be a good approximation of the data. In contrast, discretization procedures are not bound by a specific parametric distribution; yet they suffer from the obvious loss of information. Of course, one might argue that for specific tasks, such as classification, it suffices to estimate the probability that the data falls in a certain range, in which case discretization is an appropriate strategy.

In this paper, we introduce an innovative approach for dealing with continuous attributes that avoids a commitment to either one of the strategies outlined above. This approach uses a dual representation for each continuous attribute: one discretized, and the other based on fitting a parametric distribution. We use Bayesian networks to model the interaction between the discrete and continuous versions of the attribute. Then, we let the learning procedure decide which type of representation best models the training data and what interdependencies between attributes are appropriate. Thus, if attribute B can be modeled as a linear Gaussian depending on A, then the network would have a direct edge from A to B. On the other hand, if the parametric family cannot fit the dependency of B on A, then the network might use the discretized representation of A and B to model this relation. Note that the resulting models can (and usually do) involve both parametric and discretized models of interactions among attributes.

In this paper we focus our attention on classification tasks. We extend a Bayesian network classifier, introduced by Friedman, Geiger, and Goldszmidt (FGG) [8] called "Tree Augmented Naive Bayes" (TAN). FGG show that TAN outperforms naive Bayes, yet at the same time maintains the computational simplicity (no search involved) and robustness that characterize naive Bayes. They tested TAN on problems from the UCI repository [16], and compared it to C4.5, naive Bayes, and wrapper methods for feature selection with good results. The original version of TAN is restricted to multinomial distributions and discrete attributes. We start by extending the set of distributions that can be represented in TAN to include Gaussians, mixtures of Gaussians, and linear models. This extension results in classifiers that can deal with a combination of discrete and continuous attributes and model interactions between them. We compare these classifiers to the original TAN on several UCI data sets. The results show that neither approach dominates the other in terms of classification accuracy.

We then augment TAN with the capability of representing

*Current address: Institute of Computer Science, The Hebrew University, Givat Ram, Jerusalem 91904, Israel, nir@cs.huji.ac.il.

each continuous attribute in both parametric and discretized forms. We examine the consequences of the dual representation of such attributes, and characterize conditions under which the resulting classifier is well defined. Our main hypothesis is that the resulting classifier will usually achieve classification performance that is as good or better than both the purely discrete and purely continuous TAN models. This hypothesis is supported by our experiments.

We note that this dual representation capability also has ramifications in tasks such as density estimation, clustering, and compression, which we are currently investigating and some of which we discuss below. The extension of the dual representation to arbitrary Bayesian networks, and the extension of the discretization approach introduced by Friedman and Goldszmidt [9] to take the dual representation into account, are the subjects of current research.

2 REVIEW OF TAN

In this discussion 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 $\mathbf{x}, \mathbf{y}, \mathbf{z}$.

A *Bayesian network* over a set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ is an annotated directed acyclic graph that encodes a joint probability distribution over $\mathbf{X}$. Formally, a Bayesian network is a pair $B = \langle G, \mathcal{L} \rangle$. 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 second component of the pair, namely $\mathcal{L}$, represents a set of local *conditional probability distributions* (CPDs) $L_1, \ldots, L_n$, where the CPD for X_i maps possible values x_i of X_i and $\mathbf{pa}(X_i)$ of $\mathbf{Pa}(X_i)$, the set of parents of X_i in G, to the conditional probability (density) of x_i given $\mathbf{pa}(X_i)$. A Bayesian network B defines a unique joint probability distribution (density) over $\mathbf{X}$ given by the product

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} L_i(X_i|\mathbf{Pa}(X_i)) \,. \qquad (1)$$

When the variables in $\mathbf{X}$ take values from finite discrete sets, we typically represent CPDs as tables that contain parameters $\theta_{x_i|\mathbf{pa}(X_i)}$ for all possible values of X_i and $\mathbf{Pa}(X_i)$. When the variables are continuous, we can use various parametric and semiparametric representations for these CPDs.

As an example, let $\mathbf{X} = \{A_1, \ldots, A_n, C\}$, 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, $\mathbf{Pa}(C) = \emptyset$, and each attribute has the class variable as its unique parent, namely, $\mathbf{Pa}(A_i) = \{C\}$ for all $1 \le i \le n$. 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 the definition of the *naive Bayesian classifier* commonly found in the literature [5].

The naive Bayesian classifier has been used extensively for classification. It has the attractive properties of being robust and easy to learn—we only need to estimate the CPDs $\Pr(C)$ and $\Pr(A_i \mid C)$ for all attributes. Nonetheless, the naive Bayesian classifier embodies the strong independence assumption that, given the value of the class, attributes are independent of each other. FGG [8] suggest the removal of these independence assumptions by considering a richer class of networks. They define the TAN Bayesian classifier that learns a network in which each attribute has the class and at most one other attribute as parents. Thus, the dependence among attributes in a TAN network will be represented via a tree structure. Figure 1 shows an example of a TAN network.

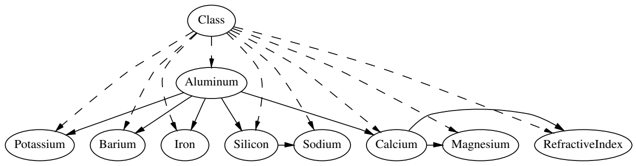


Figure 1: A TAN model learned for the data set "glass2." The dashed lines represent edges required by the naive Bayesian classifier. The solid lines are the tree augmenting edges representing correlations between attributes.

In a TAN network, an edge from A_i to A_j implies that the influence of A_i on the assessment of the class also depends on the value of A_j. For example, in Figure 1, the influence of the attribute "Iron" on the class C depends on the value of "Aluminum," while in the naive Bayesian classifier the influence of each attribute on the class is independent of other attributes. These edges affect the classification process in that a value of "Iron" that is typically surprising (i.e., $P(i|c)$ is low) may be unsurprising if the value of its correlated attribute, "Aluminum," is also unlikely (i.e., $P(i|c,a)$ is high). In this situation, the naive Bayesian classifier will overpenalize the probability of the class by considering two unlikely observations, while the TAN network of Figure 1 will not do so, and thus will achieve better accuracy.

TAN networks have the attractive property of being learnable in polynomial time. FGG pose the learning problem as a search for the TAN network that has the highest *likelihood* $LL(B : D) = P_B(D)$, given the data D. Roughly speaking, networks with higher likelihood match the data better. FGG describe a procedure `Construct-TAN` for learning TAN models and show the following theorem.

Theorem 2.1: [8] *Let D be a collection of N instances of $C, A_1, \ldots, A_n$. The procedure* `Construct-TAN` *builds a TAN network B that maximizes $LL(B : D)$ and has time complexity $O(n^2 \cdot N)$.*

The TAN classifier is related to the classifier introduced by Chow and Liu [2]. That method learns a different tree for each class value. FGG's results show that the TAN and Chow and Liu's classifier perform roughly the same. In domains where there is substantial differences in the interactions between attributes for different class values, Chow and Liu's method performs better. In others, it is possible to learn a better tree by pooling the examples from different classes as done by TAN. Although we focus on extending the TAN classifier here, all of our ideas easily apply to classifiers that learn a different tree for each class value.

3 GAUSSIAN TAN

The TAN classifier, as described by FGG, applies only to discrete attributes. In experiments run on data sets with continuous attributes, FGG use the prediscretizion described by Fayyad and Irani [7] before learning a classifier. In this paper, we attempt to model the continuous attributes directly within the TAN network. To do so, we need to learn CPDs for continuous attributes. In this section, we discuss Gaussian distributions for such CPDs. The theory of training such representations is standard (see, for example, [1, 5]). We only review the indispensable concepts.

A more interesting issue pertains to the structure of the network. As we shall see, when we mix discrete and continuous attributes, the algorithms must induce directed trees. This is in contrast to the procedure of FGG, which learns undirected trees and then arbitrarily chooses a root to define edge directions. We describe the procedure for inducing directed trees next.

3.1 THE BASIC PROCEDURE

We now extend the TAN algorithm for directed trees. This extension is fairly straight forward and similar ideas have been suggested for learning tree-like Bayesian networks [12]. For completeness, and to facilitate later extensions, we rederive the procedure from basic principles. Assume that we are given a data set D that consists of N identically and independently distributed (i.i.d.) instances that assign values to $A_1, \ldots, A_n$ and C. Also assume that we have specified the class of CPDs that we are willing to consider. The objective is, as before, to build a network that maximizes the *likelihood* function $LL(B : D) = \log P_B(D)$.

Using Eq. (1) and the independence of training instances, it is easy to show that

$$\begin{aligned} LL(B : D) &= \textstyle\sum_i \sum_{j=1}^{N} \log L_i(x_i^j \mid \mathbf{Pa}(X_i)^j) \\ &= \textstyle\sum_i S(X_i \mid \mathbf{Pa}(X_i) : L_i), \end{aligned} \quad (2)$$

where x_i^j and $\mathbf{Pa}(X_i)^j$ are the values of X_i and $\mathbf{Pa}(X_i)$ in the jth instance in D. We denote by $S(X_i \mid \mathbf{Pa}(X_i))$ the value attained by $S(X_i \mid \mathbf{Pa}(X_i), L_i)$ when L_i is the optimal CPD for this family, given the data, and the set of CPDs we are willing to consider (e.g., all tables, or all Gaussian distributions). "Optimal" should be understood in terms of maximizing the likelihood function in Eq. (2).

We now recast this decomposition in the special class of TAN networks. Recall that in order to induce a TAN network, we need to choose for each attribute A_i at most one parent other than the class C. We represent this selection by a function $\pi(i)$, s.t., if $\pi(i) = 0$, then C is the only parent of A_i, otherwise both $A_{\pi(i)}$ and C are the parents of A_i. We define $LL(\pi : D)$ to be the likelihood of the TAN model specified by π, where we select an optimal CPD for each parent set specified by π. Rewriting Eq. (2), we get

$$\begin{aligned} & LL(\pi : D) \\ &= \textstyle\sum_{i,\pi(i)>0} S(A_i \mid C, A_{\pi(i)}) + \\ &\quad \textstyle\sum_{i,\pi(i)=0} S(A_i \mid C) + S(C \mid \emptyset) \\ &= \textstyle\sum_{i,\pi(i)>0} (S(A_i \mid C, A_{\pi(i)}) - S(A_i \mid C)) + \\ &\quad \textstyle\sum_i S(A_i \mid C) + S(C \mid \emptyset) \\ &= \textstyle\sum_{i,\pi(i)>0} (S(A_i \mid C, A_{\pi(i)}) - S(A_i \mid C)) + c, \end{aligned}$$

where c is some constant that does not depend on π. Thus, we need to maximize only the first term. This maximization can be reduced to a graph-theoretic maximization by the following procedure, which we call `Directed-TAN`:

1. Initialize an empty graph $\mathcal{G}$ with n vertices labeled $1, \ldots, n$.
2. For each attribute A_i, find the best scoring CPD for $P(A_i \mid C)$ and compute $S(A_i \mid C)$. For each A_j with $j \neq i$, if an arc from A_j to A_i is legal, then find the best CPD for $P(A_i \mid C, A_i)$, compute $S(A_i \mid C, A_j)$, and add to $\mathcal{G}$ an arc $j \rightarrow i$ with weight $S(A_i \mid C, A_j) - S(A_i \mid C)$.
3. Find a set of arcs $\mathcal{A}$ that is a *maximal weighted branching* in $\mathcal{G}$. A branching is a set of edges that have at most one member pointing into each vertex and does not contain cycles. Finding a maximally weighed branching is a standard graph-theoretic problem that can be solved in low-order polynomial time [6, 17].
4. Construct the TAN model that contains arc from C to each A_i, and arc from A_j to A_i if $j \rightarrow i$ is in $\mathcal{A}$. For each A_i, assign it the best CPD found in step 2 that matches the choice of arcs in the branching.

From the arguments we discussed above it is easy to see that this procedure constructs the TAN model with the highest score. We note that since we are considering directed edges, the resulting TAN model might be a forest of directed trees instead of a spanning tree.

Theorem 3.1: *The procedure* `Directed-TAN` *constructs a TAN network B that maximizes $LL(B : D)$ given the constraints on the CPDs in polynomial time.*

In the next sections we describe how to compute the optimal S for different choices of CPDs that apply to different types of attributes.

3.2 DISCRETE ATTRIBUTES

Recall that if A_i is discrete, then we model $P(A_i \mid \mathbf{Pa}(A_i))$ by using tables that contain a parameter $\theta_{a_i \mid \mathbf{pa}(A_i)}$ for each choice of values for A_i and its parents. Thus,

$$\begin{aligned} S(A_i \mid \mathbf{Pa}(A_i)) &= \sum_j \log P(a_i^j \mid \mathbf{pa}(A_i)^j) \\ &= N \sum_{a_i, \mathbf{pa}(A_i)} \hat{P}(a_i, \mathbf{pa}(A_i)) \log \theta_{a_i \mid \mathbf{pa}(A_i)}, \end{aligned}$$

where $\hat{P}(\cdot)$ is the empirical frequency of events in the training data. Standard arguments show that the maximum likelihood choice of parameters is $P(x \mid y) = \hat{P}(x \mid y)$. Making the appropriate substitution above, we get a nice information-theoretic interpretation of the weight of the arc from A_i to A_j, $S(A_i \mid C, A_j) - S(A_i \mid C) = N \cdot I(A_i; A_j \mid C)$. The $I()$ term is the *conditional mutual information* between A_i and A_j given C [3]. Roughly speaking, it measures how much information A_j provides about A_i if already know the value of C. In this case, our procedure reduces to `Construct-TAN` of FGG, except that they use $I(A_i; A_j \mid C)$ directly as the weight on the arcs, while we multiply these weights by N.

3.3 CONTINUOUS ATTRIBUTES

We now consider the case where X is continuous. There are many possible parametric models for continuous variables. Perhaps the easiest one to use is the Gaussian distribution. A continuous variable is a Gaussian with mean μ and variance σ^2 if the pdf of X has the form $\varphi(x : \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. If all the parents of a continuous A_i are discrete, then we learn a conditional Gaussian CPD [11, 15] by assigning to A_i different mean $\mu_{a_i|\mathbf{pa}(A_i)}$ and variance $\sigma^2{}_{a_i|\mathbf{pa}(A_i)}$ for each joint value of its parents. Standard arguments (e.g., see [1]) show that we can rewrite $S(A_i \mid \mathbf{Pa}(A_i))$ as a function of $E[A_i \mid \mathbf{pa}(A_i)]$ and $E[A_i^2 \mid \mathbf{pa}(A_i)]$—the expectations of A_i and A_i^2 in these instances of the data where $\mathbf{Pa}(A_i)$ take a particular value. Standard arguments also show that we maximize the likelihood score of the CPD by choosing

$$
\begin{aligned}
\mu_{A_i|\mathbf{pa}(A_i)} &= E[A_i \mid \mathbf{pa}(A_i)] \\
\sigma^2{}_{A_i|\mathbf{pa}(A_i)} &= E[A_i^2 \mid \mathbf{pa}(A_i)] - E^2[A_i \mid \mathbf{pa}(A_i)] \ .
\end{aligned}
$$

When we learn TAN models in domains with many continuous attributes, we also want to have families where one continuous attribute is a parent of another continuous attribute. In the Gaussian model, we can represent such CPDs by using a linear Gaussian relation. In this case, the mean of A_i depends, in a linear fashion, on the value of A_j. This relationship is parameterized by three parameters: $\alpha_{A_i|A_j,c}, \beta_{A_i|A_j,c}$ and $\sigma^2{}_{A_i|A_j,c}$ for each value c of the class variable. The conditional probability for this CPD is a Gaussian with mean $\alpha_{A_i|A_j,C} + A_j \beta_{A_i|A_j,C}$ and variance $\sigma^2{}_{A_i|A_j,C}$. Again, by using standard arguments, it is easy to show that $S(A_i \mid A_j, C)$ is a function of low-order statistics in the data, and that the maximal likelihood parameters are

$$
\begin{aligned}
\beta_{A_i|A_j,c} &= \frac{E[A_iA_j|c] - E[A_i|c]E[A_j|c]}{E[A_j^2|c] - E^2[A_j|c]} \\
\alpha_{A_i|A_j,c} &= E[A_i \mid c] - \beta_{A_i|A_j,c} * E[A_j \mid c] \\
\sigma^2{}_{A_i|A_j,c} &= E[A_i^2 \mid c] - E^2[A_i \mid c] - \\
&\quad \frac{(E[A_iA_j|c] - E[A_i|c]E[A_j|c])^2}{E[A_j^2|c] - E^2[A_j|c]}
\end{aligned}
$$

In summary, to estimate parameters and to evaluate the likelihood, we need only to collect the statistics of each pair of attributes with the class, that is, terms of the form $E[A_i \mid a_j, c]$ and $E[A_iA_j \mid c]$. Thus, learning in the case of continuous Gaussian attributes can be done efficiently in a single pass over the data.

When we learn TAN models that contain discrete and Gaussian attributes, we restrict ourselves to arcs between discrete attributes, arcs between continuous attributes, and arcs from discrete attributes to continuous ones. If we want also to model arcs from continuous to discrete, then we need to introduce additional types of parametric models, such as logistic regression [1]. As we will show, an alternative solution is provided by the dual representation approach introduced in this paper.

3.4 SMOOTHING

One of the main risks in parameter estimation is overfitting. This can happen when the parameter in question is learned from a very small sample (e.g., predicting A_i from values of A_j and of C that are rare in the data). A standard approach to this problem is to *smooth* the estimated parameters. Smoothing ensures that the estimated parameters will not be overly sensitive to minor changes in the training data. FGG show that in the case of discrete attributes, smoothing can lead to dramatic improvement in the performance of the TAN classifier. They use the following smoothing rule for the discrete case

$$
\theta_{a_i|\mathbf{pa}(A_i)} = \frac{N \cdot \hat{P}(\mathbf{pa}(A_i))\hat{P}(a_i|\mathbf{pa}(A_i)) + s \cdot \hat{P}(a_i)}{N \cdot \hat{P}(\mathbf{pa}(A_i)) + s}
$$

where s is a parameter that controls the magnitude of the smoothing (FGG use $s = 5$ in all of their experiments.) This estimate uses a linear combination of the maximum likelihood parameters and the unconditional frequency of the attribute. It is easy to see that this prediction biases the learned parameters in a manner that depends on the weight of the smoothing parameter and the number of "relevant" instances in the data. This smoothing operation is similar to (and motivated by) well-known methods in statistics such as *hierarchical Bayesian* and *shrinkage* methods [10].

We can think of this smoothing operation as pretending that there are s additional instances in which A_j is distributed according to its marginal distribution. This immediately suggests how to smooth in the Gaussian case: we pretend that for these additional s samples A_i, A_i^2 have the same average as what we encounter in the totality of the training data. Thus, the statistics from the augmented data are

$$
\begin{aligned}
\hat{E}[A_i \mid \mathbf{pa}(A_i)] &= \frac{N \cdot \hat{P}(\mathbf{pa}(A_i))E[A_i|\mathbf{pa}(A_i)] + s \cdot E[A_i]}{N \cdot \hat{P}(\mathbf{pa}(A_i)) + s} \\
\hat{E}[A_i^2 \mid \mathbf{pa}(A_i)] &= \frac{N \cdot \hat{P}(\mathbf{pa}(A_i))E[A_i^2|\mathbf{pa}(A_i)] + s \cdot E[A_i^2]}{N \cdot \hat{P}(\mathbf{pa}(A_i)) + s}
\end{aligned}
$$

We then use these adjusted statistics for estimating the mean and variance of A_i given its parents. The same basic smoothing method applies for estimating linear interactions between continuous attributes.

4 SEMIPARAMETRIC ESTIMATION

Parametric estimation methods assume that the data is (approximately) distributed according to a member of the given parametric family. If the data behaves differently enough, then the resulting classifier will degrade in performance. For example, suppose that for a certain class c, the attribute A_i has bimodal distribution, where the two modes x_1 and x_2 are fairly far apart. If we use a Gaussian to estimate the distribution of A_i given C, then the mean of the Gaussian would be in the vicinity of $\mu = \frac{x_1+x_2}{2}$. Thus, instances where A_i has a value near μ would receive a high probability, given the class c. On the other hand, instances where A_i has a value in the vicinity of either x_1 or x_2 would receive a much lower probability given c. Consequently, the support c gets from A_i behaves exactly the opposite of the way it should. It is not surprising that in our experimental results, Gaussian TAN occasionally performed much worse than the discretized version (see Table 1).

A standard way of dealing with such situations is to allow the classifier more flexibility in the type of distributions it learns. One approach, called semiparametric estimation, learns a collection of parametric models. In this approach, we model $P(A_i \mid \mathbf{Pa}(A_i))$ using a mixture of Gaussian distributions: $P(A_i \mid \mathbf{pa}(A_i)) = \sum_j \varphi(A_i :$

$\mu_{A_i|\mathbf{pa}(A_i),j}, \sigma^2{}_{A_i|\mathbf{pa}(A_i),j}) w_{A_i|\mathbf{pa}(A_i),j}$, where the parameters specify the mean and variance of each Gaussian in the mixture and $w_{A_i|\mathbf{pa}(A_i),j}$ are the weights of the mixture components. We require that the $w_{A_i|\mathbf{pa}(A_i),j}$ sum up to 1, for each value of $\mathbf{Pa}(A_i)$.

To estimate $P(A_i \mid \mathbf{pa}(A_i))$, we need to decide on the number of mixture components (the parameter j in the equation above) and on the best choice of parameters for that mixture. This is usually done in two steps. First, we attempt to fit the best parameters for different number of components (e.g., $j = 1, 2, \ldots$), and then select an instantiation for j based on a performance criterion.

Because there is no closed form for learning the parameters we need to run a search procedure such as the Expectation-Maximization (EM) algorithm. Moreover, since EM usually finds local maxima, we have to run it several times, from different initial points, to ensure that we find a good approximation to the best parameters. This operation is more expensive than parametric fitting, since the training data cannot be summarized for training the mixture parameters. Thus, we need to perform many passes over the training data to learn the parameters. Because of space restrictions we do not review the EM procedure here, and refer the reader to [1, pp. 65–73].

With regard to selecting the number of components in the mixture, it is easy to see that a mixture with $k+1$ components can easily attain the same or better likelihood as any mixture with k components. Thus, the likelihood (of the data) alone is not a good performance criterion for selecting mixture components, since it always favors models with a higher number of components, which results in overfitting. Hence, we need to apply some form of model selection. The two main approaches to model selection are based on cross-validation to get an estimate of true performance for each choice of k, or on penalizing the performance on the training data to account for the complexity of the learned model. For simplicity, we use the latter approach with the BIC/MDL penalization. This rule penalizes the score of each mixture with $\frac{\log N}{2} 3k$, where k is the number of mixture components, and N is the number of training examples for this mixture (i.e., the number of instances in the data with this specific value of the discrete parents).

Once more, smoothing is crucial for avoiding overfitting. Because of space considerations we will not go into the details. Roughly speaking, we apply the Gaussian smoothing operation described above in each iteration of the EM procedure. Thus, we assume that each component in the mixture has a preassigned set of s samples it has to fit.

As our experimental results show, the additional flexibility of the mixture results in drastically improved performance in the cases where the Gaussian TAN did poorly (see, for example, the accuracy of the data sets "anneal-U" and "balance-scale" in Table 1). In this paper, we learned mixtures only when modeling a continuous feature with discrete parents. We note, however, that learning a mixture of linear models is a relatively straightforward extension that we are currently implementing and testing.

5 DUAL REPRESENTATION

The classifiers we have presented thus far require us to make a choice. We can either prediscretize the attributes and use the discretized TAN, or we can learn a (semi)parametric density model for the continuous attributes. Each of these methods has its advantages and problems: Discretization works well with nonstandard densities, but clearly loses much information about the features. Semiparametric estimation can work well for "well-behaved" multimodal densities. On the other hand, although we can approximate any distribution with a mixture of Gaussians, if the density is complex, then we need a large number of training instances to learn a mixture with large number of components, with sufficient confidence.

The choice we are facing is not a simple binary one, that is, to discretize or not to discretize all the attributes. We can easily imagine situations in which some of several attributes are better modeled by a semiparametric model, and others are better modeled by a discretization. Thus, we can choose to discretize only a subset of the attributes. Of course, the decision about one attribute is not independent of how we represent other attributes. This discussion suggests that we need to select a subset of variables to discretize, that is, to choose from an exponential space of options.

In this section, we present a new method, called *hybrid TAN*, that avoids this problem by representing both the continuous attributes and their discretized counterparts within the same TAN model. The structure of the TAN model determines whether the interaction between two attributes is best represented via their discretized representation, their continuous representation, or a hybrid of the discrete representation of one and the continuous representation of the other. Our hypothesis is that hybrid TAN allows us to achieve performance that is as good as either alternative. Moreover, the cost of learning hybrid TAN is about the same as that of learning either alternative.

Let us assume, that the first k attributes, $A_1, \ldots, A_k$, are the continuous attributes in our domain. We denote by $A_1^*, \ldots, A_k^*$ the corresponding discretized attributes (i.e., A_1^* is the discretized version of A_1), based on a predetermined discretization policy (e.g., using a standard method, such as Fayyad and Irani's [7]). Given this semantics for the discretized variables, we know that that each A_i^* is a *deterministic* function of A_i. That is, A_i^* state corresponds to the interval $[x_1, x_2]$ if and only if $A_i \in [x_1, x_2]$. Thus, even though the discretized variables are not observed in the training data, we can easily augment the training data with the discretized version of each continuous attribute.

At this stage one may consider the application of one of the methods we described above to the augmented training set. This, however, runs the risk of "double counting" the evidence for classification provided by the duplicated attributes. The likelihood of the learned model will contain a penalty for both the continuous and the discrete versions of the attribute. Consequently, during classification, a "surprising" value of an attribute would have twice the (negative) effect on the probability of the class variable. One could avoid this problem by evaluating only the likelihood assigned to the continuous version of the attributes. Unfortunately, in this case the basic decomposition of Eq. (2) no longer holds, and we cannot use the TAN procedure.

5.1 MODELING THE DUAL REPRESENTATION

Our approach takes advantage of Bayesian networks to model the interaction between an attribute and its discretized version. We constrain the networks we learn to match our model of the discretization, that is, a discretized attribute is a function of the continuous one. More specifically, for each continuous attribute A_i, we require that $P_B(A_i^* \mid A_i) = 1$ iff A_i is in the range specified by A_i^*. It is easy to show (using the chain rule) that this constraint implies that $P_B(A_1, \ldots, A_n, A_1^*, \ldots, A_k^*) = P_B(A_1, \ldots, A_n)$ avoiding the problem outlined in the previous paragraph.

Note that by imposing this constraint we are not requiring in any way that A_i be a parent of A_i^*. However, we do need to ensure that $P(A_i^* \mid A_i)$ is deterministic in the learned model. We do so by requiring that A_i and A_i^* are adjacent in the graph (i.e., one is the parent of the other) and by putting restrictions on the models we learn for $P(A_i \mid A_i^*)$ and $P(A_i^* \mid A_i)$. There are two possibilities:

If $A_i \rightarrow A_i^*$ is in the graph, then the conditional distribution $P(A_i^* \mid A_i, C)$ is determined as outlined above; it is 1 if A_i is in the range defined by the value of A_i^* and 0 otherwise.

If $A_i^* \rightarrow A_i$ is in the graph, then we require that $P(A_i \mid A_i^*, C) = 0$ whenever A_i is not in the range specified by A_i^*. By Bayes rule $P(A_i^* \mid A_i) \propto \sum_C P(A_i \mid A_i^*, C) P(A_i^*, C)$; Thus, if A_i is not in the range of A_i^*, then $P(A_i^* \mid A_i) \propto \sum_C 0 \times P(A_i^*, C) = 0$. Since the conditional probability of A_i^* given A_i must sum to 1, we conclude that $P(A_i^* \mid A_i) = 1$ iff A_i is in the range of A_i^*.

There is still the question of the form of $P(A_i \mid A_i^*, C)$. Our proposal is to learn a model for A_i given A_i^* and C, using the standard methods above (i.e., a Gaussian or a mixture of Gaussians). We then truncate the resulting density on the boundaries of the region specified by the discretization, and we ensure that the truncated density has total mass 1 by applying a normalizing constant. In other words, we learn an unrestricted model, and then condition on the fact that A_i can only take values in the specified interval.

Our goal is then to learn a TAN model that includes both the continuous and discretized versions of each continuous attribute, and that satisfies the restrictions we just described. Since these restrictions are not enforced by the procedure of Section 3.1, we need to augment it. We start by observing that our restrictions imply that if we include $B \rightarrow A$ in the model, we must also include $A \rightarrow A^*$. To see this, note that since A already has one parent (B) it cannot have additional parents. Thus, the only way of making A and A^* adjacent is by adding the edge $A \rightarrow A^*$. Similarly, if we include the edge $B \rightarrow A^*$, we must also include $A^* \rightarrow A$.

This observation suggests that we consider edges between *groups* of variables, where each group contains both versions of an attribute. In building a TAN structure that includes both representations, we must take into account that adding an edge to an attribute in a group, immediately constraints the addition of other edges within the group. Thus, the TAN procedure should make choices at the level groups. Such a procedure, which we call *hybrid-TAN* is described next.

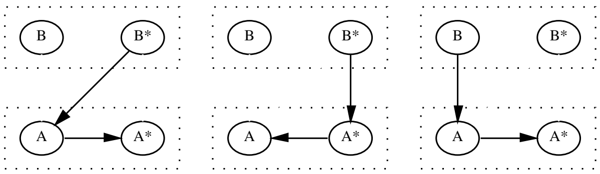


Figure 2: The three possible ways of placing an edge from $\{B, B^*\}$ into $\{A, A^*\}$. The parameterization of possible arcs are as follows: $B^* \rightarrow A^*$ is a discrete model, both $B^* \rightarrow A$ and $B \rightarrow A$ are continuous models (e.g., Gaussians), $A^* \rightarrow A$ is a truncated continuous model (e.g., truncated Gaussian), and $A \rightarrow A^*$ is a deterministic model.

5.2 HYBRID-TAN

We now expand on the details of the procedure. As with the basic procedure, we compute scores on edges. Now, however, edges are between groups of attributes. Each group consisting of the different representations of an attribute.

Let A be a continuous attribute. By our restriction, either $A \in \mathbf{Pa}(A^*)$, or $A^* \in \mathbf{Pa}(A)$. And since each attribute has at most one parent (in addition to the class C), we have that at most one other attribute is in $\mathbf{Pa}(A) \cup \mathbf{Pa}(A^*) - \{A, A^*, C\}$. We define a new function $T(A \mid B)$ that denotes the *best* combination of parents for A and A^* such that either B or B^* is a parent of one of these attributes. Similarly, $T(A \mid \emptyset)$ denotes the best configuration such that no other attribute is a parent of A or A^*.

First, consider the term $T(A \mid \emptyset)$. If we decide that neither A nor A^* have other parents, then we can freely choose between $A \rightarrow A^*$ and $A^* \rightarrow A$. Thus

$$\begin{aligned} T(A \mid \emptyset) = \quad \max(\quad & S(A \mid C, A^*) + S(A^* \mid C), \\ & S(A \mid C) + S(A^* \mid C, A)), \end{aligned}$$

where $S(A \mid C, A^*)$ and $S(A^* \mid C, A)$ are the scores of the CPDs subject to the constraints discussed in Subsection 5.1 (the first is a truncated model, and the second is a deterministic model).

Next, consider the case that a continuous attribute B is a parent of A. There are three possible ways of placing an edge from the group $\{B, B^*\}$ into the group $\{A, A^*\}$. These cases are shown in Figure 2. (The fourth case is disallowed, since we cannot have an edge from the continuous attribute, B to the discrete attribute, A^*.) It is easy to verify that in any existing TAN network, we can switch between the edge configurations of Figure 2 without introducing new cycles. Thus, given the decision that the group B points to the group A, we would choose the configuration with maximal score:

$$\begin{aligned} T(A \mid B) = \quad \max(\quad & S(A \mid C, B^*) + S(A^* \mid C, A), \\ & S(A \mid C, A^*) + S(A^* \mid C, B^*), \\ & S(A \mid C, B) + S(A^* \mid C, A)) \end{aligned}$$

Finally, when B is discrete, then $T(A \mid B)$ is the maximum between two options (B as a parent of A or as a parent of B^*), and when A is discrete, then $T(A \mid B)$ is equal to one term (either $S(A \mid C, B)$ or $S(A \mid C, B^*)$, depending on B's type).

We now define the `Hybrid-TAN` procedure:

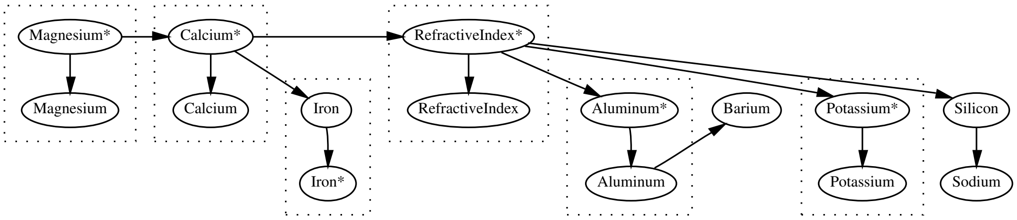


Figure 3: A hybrid TAN model learned for the data set "glass2." For clarity, the edges from the class to all the attributes are not shown. The attributes marked with asterisks ($*$) correspond to the discretized representation. Dotted boxes mark two versions of the same attribute.

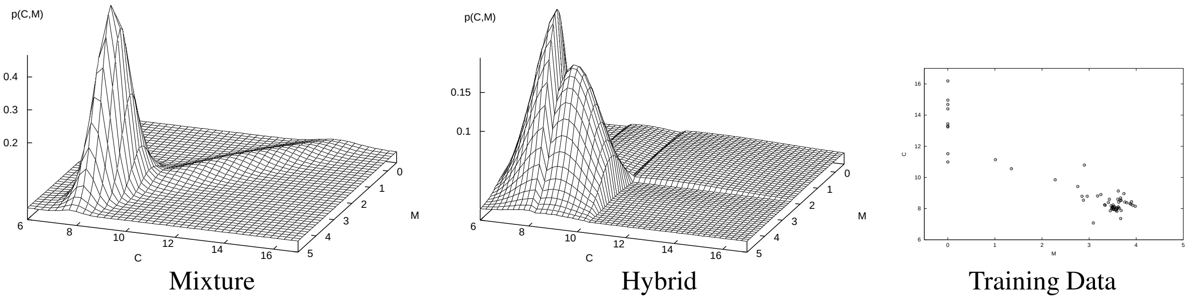


Figure 4: Differences in the modeling of the interaction between attributes, for mixtures of Gaussians and the hybrid model. The graphs show the interaction between Calcium (C) and Magnesium (M) in the "glass2" data set, given a specific value of the class.

1. Initialize an empty graph $\mathcal{G}$ with n vertices labeled $1, \ldots, n$.
2. For each attribute A_i, compute the scores of the form $S(A_i \mid C)$, $S(A_i^* \mid C)$, $S(A_i \mid C, A_i^*)$, etc. For each A_j with $j \neq i$, add to $\mathcal{G}$ an arc $j \rightarrow i$ with weight $T(A_i \mid A_j) - T(A_i \mid \emptyset)$.
3. Find a maximal weighted branching $\mathcal{A}$ in $\mathcal{G}$.
4. Construct the TAN model that contains edges from C to each A_i and A_i^*. If $j \rightarrow i$ is in $\mathcal{A}$, add the best configuration of edges (and the corresponding CPDs) from the group A_j into A_i. If i does not have an incoming arc in $\mathcal{A}$, then add the edge between A_i and A_i^* that maximizes $T(A_i : \emptyset)$.

It is straight forward to verify that this procedure performs the required optimization:

Theorem 5.1: *The procedure* `Hybrid-TAN` *constructs in polynomial time a dual TAN network B that maximizes $LL(B : D)$, given the constraints on the CPDs and the constraint that A_i and A_i^* are adjacent in the graph.*

5.3 AN EXAMPLE

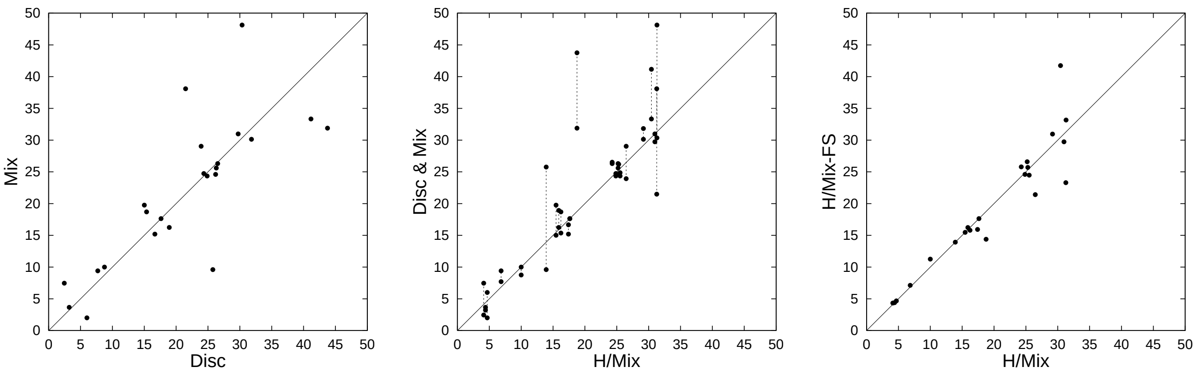
Figure 3 shows an example of a hybrid TAN model learned from one of the folds of the "glass2" data set.[1] It is instructive to compare it to the network in Figure 1, which was learned by a TAN classifier based on mixtures of Gaussians from the same data set. As we can see, there are some similarities between the networks, such as the connections between "Silicon" and "Sodium," and between "Calcium" and "Magnesium" (which was reversed in the hybrid version). However, most of the network's structure is quite different. Indeed, the relation between "Magnesium" and "Calcium" is now modulated by the discretized version of these variables. This fact, and the increased accuracy of hybrid TAN for this data set (see Table 1), seem to indicate that in this domain attributes are not modeled well by Gaussians.

As a further illustration of this, we show in Figure 4 the estimate of the joint density of "Calcium" and "Magnesium" in both networks (given a particular value for the class), as well as the training data from which both estimates were learned. As we can see, most of the training data is centered at one point (roughly, when $M = 3.5$ and $C = 8$), but there is fair dispersion of data points when $M = 0$. In the Gaussian case, C is modeled by a mixture of two Gaussians (centered on 8.3 and 11.8, where the former has most of the weight in the mixture), and M is modeled as a linear function of C with a fixed variance. Thus, we get a sharp "bump" at the main concentration point on the low ridge in Figure 4a. On the other hand, in the hybrid model, for each attribute, we model the probability in each bin by a truncated Gaussian. In this case, C is partitioned into three bins and M into two. This model results in the discontinuous density function we see in Figure 4b. As we can see, the bump at the center of concentration is now much wider, and the whole region of dispersion corresponds to a low, but wide, "tile" (in fact, this tile is a truncated Gaussian with a large variance).

6 EXPERIMENTAL EVALUATION

We ran our experiments on the 23 data sets listed in Table 1. All of these data sets are from the UCI repository [16], and are accessible at the MLC++ ftp site. The accuracy of each classifier is based on the percentage of successful pre-

[1] Some of the discrete attributes do not appear in the figure, since they were discretized into one bin.



(a) (b) (c)

Figure 5: Scatter plots comparing the performance (a) of **Disc** (x axis) vs. **Mix** (y axis), (b) of **H/Mix** (x axis) vs. **Disc** and **Mix** (y **axis**), and (c) of **H/Mix** (x axis) vs. **H/Mix-FS** (y axis). In these plots, each point represents a data set, and the coordinates correspond to the prediction error of each of the methods compared. Points below the diagonal line correspond to data sets where the y axis method is more accurate, and points above the diagonal line correspond to data sets where the x axis method is more accurate. In (b), the dashed lines connect points that correspond to the same data set.

dictions on the test sets of each data set. We estimate the prediction accuracy of each classifier as well as the variance of this accuracy by using the MLC++ system [14]. Accuracy was evaluated using 5-fold *cross validation* (using the methods described in [13]). Since we do not currently deal with missing data, we removed instances with missing values from the data sets. To construct discretizations, we used a variant of the method of Fayyad and Irani [7], using only the training data, in the manner described in [4]. These preprocessing stages were 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 exactly the same test sets.

Table 1 summarizes the accuracies of the learning procedures we have discussed in this paper: (1) **Disc**–TAN classifier based on prediscretized attributes; (2) **Gauss**–TAN classifier using Gaussians for the continuous attributes and multinomials for the discrete ones; (3) **Mix**–TAN classifier using mixtures of Gaussians for the continuous attributes; (4) **H/Gauss**–hybrid TAN classifier enabling the dual representation and using Gaussians for the continuous version of the attributes; (5) **H/Mix**–hybrid TAN classifier using mixtures of Gaussian for the continuous version of the attributes; and (6) **H/Mix-FS**–same as **H/Mix** but incorporating a primitive form of feature selection. The discretization procedure often removes attributes by discretizing them into one interval. Thus, these attributes are ignored by the discrete version of TAN. **H/Mix-FS** imitate this feature selection by also ignoring the continuous version of the attributes removed by the discretization procedure.

As we can see in Figure 5(a), neither the discrete TAN (**Disc**) nor the mixture of Gaussians TAN (**Mix**) outperforms the other. In some domains, such as "anneal-U" and "glass," the discretized version clearly performs better; in others, such as "balance-scale," "hayes-roth," and "iris," the semiparametric version performs better. Note that the latter three data sets are all quite small. So, a reasonable hypothesis is that the data is too sparse to learn good discretizations. On the other hand, as we can see in Figure 5(b), the hybrid method performs at roughly the same level as the best of either **Mix** or **Disc** approaches. In this plot, each pair of connected points describes the accuracy results achieved by **Disc** and **Mix** for a single data set. Thus, the best accuracy of these two methods is represented by the lower point on each line. As we can see, in most data sets the hybrid method performs roughly at the same level as these lower points. In addition, in some domains such as "glass2," "hayes-roth," and "hepatitis" the ability to model more complex interactions between the different continuous and discrete attributes results in a higher prediction accuracy. Finally, given the computational cost involved in using EM to fit the mixture of Gaussians we include the accuracy of **H/Gauss** so that the benefits of using a mixture model can be evaluated. At the same time, the increase in prediction accuracy due to the dual representation can be evaluated by comparing to **Gauss**.

Due to the fact that **H/Mix** increases the number of parameters that need to be fitted, feature selection techniques are bound to have a noticeable impact. This is evident in the results obtained for **H/Mix-FS** which, as mentioned above, supports a primitive form of feature selection (see Figure 5(c)). These results indicate that we may achieve better performance by incorporating a feature selection mechanism into the classifier. We leave this as a topic for future research.

7 CONCLUSIONS

The contributions of this work are twofold. First, we extend the TAN classifier to directly model continuous attributes by parametric and semiparametric methods. We use standard procedures to estimate each of the conditional distributions, and then combine them in a structure learning phase by maximizing the likelihood of the TAN model. The resulting procedure preserves the attractive properties of the original TAN classifier—we can learn the best model in polynomial time. Of course, one might extend TAN to use other para-

Table 1: Experimental Results. The first four column describe the name of the data sets, the number of continuous and discrete attributes, and the number of instances. The remaining columns report percentage classification error and std. deviations from 5-fold cross validation of the tested procedures (see text).

	Attr.			Prediction Errors					
Data set	C	D	Size	**Disc**	**Gauss**	**Mix**	**H/Gauss**	**H/Mix**	**H/Mix-FS**
anneal-U	6	32	898	**2.45 +- 1.01**	23.06 +- 3.49	7.46 +- 3.12	10.91 +- 1.79	4.12 +- 1.78	4.34 +- 1.43
australian	6	8	690	**15.36 +- 2.37**	23.77 +- 3.26	18.70 +- 4.57	17.10 +- 2.83	16.23 +- 2.38	15.80 +- 1.94
auto	15	10	159	23.93 +- 8.57	28.41 +- 10.44	29.03 +- 10.04	27.10 +- 8.12	26.47 +- 8.44	**21.41 +- 4.27**
balance-scale	4	0	625	25.76 +- 7.56	11.68 +- 3.56	**9.60 +- 2.47**	11.84 +- 3.89	13.92 +- 2.16	13.92 +- 2.16
breast	10	0	683	**3.22 +- 1.69**	5.13 +- 1.73	3.66 +- 2.13	**3.22 +- 1.69**	4.34 +- 1.10	4.32 +- 0.96
cars1	7	0	392	26.52 +- 2.64	25.03 +- 7.11	26.30 +- 4.44	25.28 +- 6.54	**24.27 +- 7.85**	25.79 +- 6.21
cleve	6	7	296	18.92 +- 1.34	17.23 +- 1.80	16.24 +- 3.97	16.24 +- 3.97	**15.89 +- 3.14**	16.23 +- 3.58
crx	6	9	653	**15.01 +- 1.90**	24.05 +- 4.44	19.76 +- 4.04	17.31 +- 1.60	15.47 +- 1.87	15.47 +- 2.09
diabetes	8	0	768	24.35 +- 2.56	25.66 +- 2.70	24.74 +- 3.74	**22.65 +- 3.21**	24.86 +- 4.06	24.60 +- 3.45
echocardiogram	6	1	107	31.82 +- 10.34	**28.23 +- 13.86**	30.13 +- 14.94	29.18 +- 14.05	29.18 +- 14.05	30.95 +- 11.25
flare	2	8	1066	**17.63 +- 4.19**	17.91 +- 4.34	**17.63 +- 4.46**	17.91 +- 4.34	**17.63 +- 4.46**	**17.63 +- 4.19**
german-org	12	12	1000	26.30 +- 2.59	25.30 +- 2.97	25.60 +- 1.39	25.70 +- 3.47	**25.20 +- 1.75**	26.60 +- 2.27
german	7	13	1000	26.20 +- 4.13	25.20 +- 2.51	**24.60 +- 1.88**	25.10 +- 2.07	25.30 +- 3.33	25.70 +- 4.40
glass	9	0	214	**30.35 +- 5.58**	49.06 +- 6.29	48.13 +- 8.12	32.23 +- 4.63	31.30 +- 5.00	33.16 +- 5.65
glass2	9	0	163	**21.48 +- 3.73**	38.09 +- 7.92	38.09 +- 7.92	34.39 +- 9.62	31.27 +- 9.63	23.30 +- 6.22
hayes-roth	4	0	160	43.75 +- 4.42	33.12 +- 11.40	31.88 +- 6.01	29.38 +- 10.73	18.75 +- 5.85	**14.38 +- 4.19**
heart	13	0	270	16.67 +- 5.56	15.56 +- 5.65	**15.19 +- 5.46**	**15.19 +- 3.56**	17.41 +- 4.65	15.93 +- 5.34
hepatitis	6	13	80	**8.75 +- 3.42**	12.50 +- 4.42	10.00 +- 3.42	12.50 +- 7.65	10.00 +- 5.59	11.25 +- 5.23
ionosphere	34	0	351	7.70 +- 2.62	9.13 +- 3.31	9.41 +- 2.98	**6.85 +- 3.27**	**6.85 +- 3.27**	7.13 +- 3.65
iris	4	0	150	6.00 +- 2.79	**2.00 +- 2.98**	**2.00 +- 2.98**	4.67 +- 1.83	4.67 +- 1.83	4.67 +- 1.83
liver-disorder	6	0	345	41.16 +- 1.94	40.29 +- 5.16	33.33 +- 4.10	36.52 +- 7.63	**30.43 +- 5.12**	41.74 +- 2.59
pima	8	0	768	24.87 +- 2.82	24.35 +- 1.45	24.35 +- 3.47	**22.92 +- 3.96**	25.52 +- 2.85	24.48 +- 2.87
post-operative	1	7	87	**29.74 +- 13.06**	34.38 +- 10.09	30.98 +- 11.64	34.38 +- 10.09	30.98 +- 11.64	**29.74 +- 13.06**

metric families (e.g., Poisson distributions) or other semi-parametric methods, (e.g., kernel-based methods). The general conclusion we draw from these extensions is that if the assumptions embedded in the parametric forms "match" the domain, then the resulting TAN classifier generalizes well and will lead to good prediction accuracy. We also note that it is straightforward to extend the procedure to select, at learning time, a parametric form from a set of parametric families.

Second, we introduced a new method to deal with different representations of continuous attributes within a single model. This method enables our model learning procedure (in this case, TAN) to automate the decision as to which representation is most useful in terms of providing information about other attributes. As we showed in our experiments, the learning procedure managed to make good decisions on these issues and achieve performance that roughly as good as both the purely discretized and the purely continuous approaches.

This method can be extended in several directions. For example, to deal with several discretizations of the same attributes in order to select the granularity of discretization that is most useful for predicting other attributes. Another direction involves adapting the discretization to the particular edges that are present in the model. As argued Friedman and Goldszmidt [9], it is possible to discretize attributes to gain the most information about the neighboring attributes. Thus, we might follow the approach in [9] and iteratively readjust the structure and discretization to improve the score. Finally, it is clear that this hybrid method is applicable not only to classification, but also to density estimation and related tasks using general Bayesian networks. We are currently pursuing these directions.

Acknowledgments

We thank Anne Urban for help with the experiments. M. Goldszmidt and T. Lee were supported in part by DARPA's High Performance Knowledge Bases program under SPAWAR contract N66001-97-C-8548. N. Friedman was supported in part by ARO under grant DAAH04-96-1-0341.

References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition*, 1995.

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

[3] T. M. Cover and J. A. Thomas. *Elements of Information Theory*, 1991.

[4] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *ICML '95*. 1995.

[5] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*, 1973.

[6] S. Even. *Graph Algorithms*, 1979.

[7] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI '93*. 1993.

[8] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[9] N. Friedman and M. Goldszmidt. Discretization of continuous attributes while learning Bayesian networks. In *ICML '96*. 1996.

[10] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. 1995.

[11] D. Heckerman and D. Geiger. Learning Bayesian networks: a unification for discrete and Gaussian domains. In *UAI '95*. 1995.

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

[13] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI '95*. 1995.

[14] R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger. MLC++: A machine learning library in C++. In *Proc. 6'th Inter. Conf. on Tools with AI*, 1994.

[15] S. L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17:31–57, 1989.

[16] P. M. Murphy and D. W. Aha. UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 1995.

[17] R. Tarjan. Finding optimal branching. *Networks*, 7:25–35, 1977.