Machine Learning, 53, 235–263, 2003

Implications of the Dirichlet Assumption for Discretization of Continuous Variables in Naive Bayesian Classifiers

CHUN-NAN HSU chunnan@iis.sinica.edu.tw
Institute of Information Science, Academia Sinica, Nankang, Taipei City 115, Taiwan

HUNG-JU HUANG hungju@cis.nctu.edu.tw
Department of Computer and Information Science, National Chiao-Tung University, Hsinchu City 300, Taiwan

TZU-TSUNG WONG tzutsung@mail.ncku.edu.tw
Institute of Information Management, National Cheng-Kung University, Tainan City 701, Taiwan

Editor: Douglas Fisher

Abstract. In a naive Bayesian classifier, discrete variables as well as discretized continuous variables are assumed to have Dirichlet priors. This paper describes the implications and applications of this model selection choice. We start by reviewing key properties of Dirichlet distributions. Among these properties, the most important one is "*perfect aggregation*," which allows us to explain why discretization works for a naive Bayesian classifier. Since perfect aggregation holds for Dirichlets, we can explain that in general, discretization can outperform parameter estimation assuming a normal distribution. In addition, we can explain why a wide variety of well-known discretization methods, such as entropy-based, ten-bin, and bin-log *l*, can perform well with insignificant difference. We designed experiments to verify our explanation using synthesized and real data sets and showed that in addition to well-known methods, a wide variety of discretization methods all perform similarly. Our analysis leads to a *lazy* discretization method, which discretizes continuous variables according to test data. The Dirichlet assumption implies that lazy methods can perform as well as *eager* discretization methods. We empirically confirmed this implication and extended the lazy method to classify set-valued and multi-interval data with a naive Bayesian classifier.

Keywords: naive Bayesian classifiers, Dirichlet distributions, perfect aggregation, continuous variables, discretization, lazy discretization, interval data

1. Introduction

Learning a naive Bayesian classifier (a.k.a. naive Bayes) (Duda & Hart, 1973; Langley & Thompson, 1992) from data is an important technique for data classification. In spite of its simplicity, the naive Bayesian classifier can perform surprisingly well in some domains. Domingos and Pazzani (1997) provided an interesting analysis of why the naive Bayesian classifier can perform well even though it violates the independence assumption. They reported an experiment that compared the naive Bayesian classifier with several classical learning algorithms such as C4.5 with a large ensemble of real data sets. The results show that statistically the naive Bayesian classifier can outperform those algorithms significantly

in many domains. But a more interesting aspect of their result is that it was achieved by a version of the naive Bayesian classifier that uses a very "naive" discretization method to handle continuous data. That method, usually referred to as "ten-bin," divides the domain of each continuous variable into ten equal-width bins. Such a simple method was considered to be inferior and many complex methods have been proposed and applied, but surprisingly, simple discretization methods perform well while not much improvement is found when complex methods are used. Dougherty, Kohavi, and Sahami (1995) conducted an empirical study comparing the performance of four well-known discretization methods and concluded that the discretization methods can outperform a version of the naive Bayesian classifier that assumes normal distributions for continuous variables. Moreover, these discretization methods performed approximately the same, though an entropy-based method (Fayyad & Irani, 1993) performed slightly better for some data sets. Kohavi and Sahami (1996) compared the entropy-based method with an error-based method, which is not considered in Dougherty, Kohavi, and Sahami (1995). Again, both methods outperformed the normal version whereas no statistically significant difference between their performance was found.

Previous work provided some informal discussion but none explained why these discretization methods can work well regardless of their complexities. In this paper, we present an explanation of this phenomenon. The key of our explanation is the "*perfect aggregation*" property of Dirichlet distributions. In a naive Bayesian classifier, the class conditional probability of a discrete variable is usually modeled as having a multinomial distribution with a Dirichlet prior (Monti & Cooper, 1999). This model selection choice is also applied to discretized continuous variables. Perfect aggregation of Dirichlets implies that, with sufficient training examples, we can estimate the class conditional probabilities of discretized intervals with an arbitrary accuracy. This explains why a naive Bayesian classifier with discretization can be effective in approximating a broad range of probability distributions of continuous data. We also found that to achieve optimal performance, the probability corresponding to an interval must simulate the role of the true density in the classification. This led us to identify situations where discretization may cause performance degradation. We show that these situations are unlikely to happen under zero-one loss for a wide variety of discretization methods including the well-known methods. Therefore, these methods will produce similar performance for many data sets regardless of their complexities.

The properties of Dirichlet distributions implies that it is not necessary to determine cut points of the domain of a continuous variable before a naive Bayesian classifier performs classification, as in existing discretization methods. Instead, a "*lazy*" discretization method that dynamically discretizes continuous variables according to test data can achieve the same result. In this paper, we present an instance of lazy discretization method and conduct experiments to evaluate its performance. According to the Dirichlet assumption, this new method should perform equally well as the well-known methods. Experiments revealed results consistent with our prediction. Though the lazy method is not superior to existing methods, one of its contributions is in justifying the selection of Dirichlet priors and verifying our analysis. Another contribution is that the lazy discretization method can be extended to allow a naive Bayesian classifier to classify set-valued and multi-interval data.

The remainder of this paper is organized as follows. Section 2 reviews the definition and the perfect aggregation property of Dirichlet distributions. Section 3 describes the Dirichlet

assumption made by a naive Bayesian classifier and discretization methods. Section 4 explains why well-known discretization methods can outperform parameter estimation with approximately the same results. Section 5 provides empirical evidence of our explanation. Section 6 presents the lazy discretization method. Section 7 describes the use of lazy discretization for set-valued and multi-interval data. The last section summarizes the conclusions and presents directions of future research.

2. Dirichlet distributions and perfect aggregation

In a naive Bayesian classifier, discrete variables as well as discretized continuous variables are assumed to have Dirichlet priors. This section reviews the important properties of Dirichlet distributions.

2.1. Dirichlet distributions

A Dirichlet distribution is formally defined as follows (Wilks, 1962):

Definition 1. Random vector $\Theta = (\theta_1, \theta_2, \ldots, \theta_k)$ has a *k-variate Dirichlet distribution* with parameters $\alpha_j > 0$ for $j = 1, 2, \ldots, k+1$ if it has density

$$f(\Theta) = \frac{\Gamma\left(\sum_{j=1}^{k+1} \alpha_j\right)}{\prod_{j=1}^{k+1} \Gamma(\alpha_j)} \prod_{j=1}^{k} \theta_j^{\alpha_j - 1} (1 - \theta_1 - \cdots - \theta_k)^{\alpha_{k+1} - 1}$$

for $\theta_1 + \theta_2 + \cdots + \theta_k \leq 1$ and $\theta_j \geq 0$ for $j = 1, 2, \ldots, k$. This distribution will be denoted $D_k(\alpha_1, \alpha_2, \ldots \alpha_k; \alpha_{k+1})$. A *Beta distribution* is a univariate Dirichlet distribution and usually denoted $\text{Beta}(\alpha_1, \alpha_2)$.

The expected value for θ_j is $E[\theta_j] = \frac{\alpha_j}{\alpha}$, for $j = 1, 2, \ldots, k$, where $\alpha = \alpha_1 + \alpha_2 + \cdots + \alpha_{k+1}$.

An important property of Dirichlets is that the Dirichlet distribution is *conjugate* to the multinomial sampling (Wilks, 1962). This property basically states that the posterior distribution of a Dirichlet given our observation is also a Dirichlet. Formally, let $D = \{y_1, y_2, \ldots, y_{k+1}\}$ be a data set for the outcomes in n trials, where y_j denotes the number of trials that have outcome j. When the prior distribution $p(\Theta)$ is a Dirichlet distribution with parameters α_j for $j = 1, 2, \ldots, k+1$, and the likelihood function $L(D \mid \Theta)$ follows a multinomial distribution, then the posterior distribution $p(\Theta \mid D)$ is also a Dirichlet distribution with parameters $\alpha'_j = \alpha_j + y_j$ for $j = 1, 2, \ldots, k+1$. Similarly, the Beta distribution is conjugate to the binomial sampling. Due to the conjugate property, given a data set D, the expected value for θ_j will be updated and become $E[\theta_j \mid D] = \frac{\alpha_j + y_j}{\alpha + n}$, for $j = 1, 2, \ldots, k+1$.

The following "closure properties" of Dirichlet distributions are critical to our analysis. These properties greatly simplify the computation of the moments of the Dirichlet distribution in Bayesian analysis.

Lemma 1. *Suppose random vector* $\Theta = (\theta_1, \theta_2, \ldots, \theta_k)$ *has a Dirichlet distribution* $D_k(\alpha_1, \alpha_2, \ldots, \alpha_k; \alpha_{k+1})$, *then any subvector* $(\theta_{n1}, \theta_{n2}, \ldots, \theta_{nm})$ *of* θ *has an* m*-variate Dirichlet distribution* $D_m(\alpha_{n1}, \alpha_{n2}, \ldots, \alpha_{nm}; \alpha - \sum_{j=1}^{m} \alpha_{nj})$.

Lemma 2. *Suppose random vector* $\Theta = (\theta_1, \theta_2, \ldots, \theta_k)$ *has a Dirichlet distribution* $D_k(\alpha_1, \alpha_2, \ldots, \alpha_k; \alpha_{k+1})$, *then the sum of any subset* $\Phi \subseteq \{\theta_1, \theta_2, \ldots, \theta_k\}$ *has a Beta distribution with parameters* $\sum_{j\in\Phi} \alpha_j$ *and* $\alpha - \sum_{j\in\Phi} \alpha_j$, *where* $\alpha = \alpha_1 + \alpha_2 + \cdots + \alpha_{k+1}$.

The proofs of Lemmas 1 and 2 can be found in Wilks (1962).

2.2. *Perfect aggregation*

Let Φ be a subset of $\{\theta_1, \theta_2, \ldots, \theta_k\}$, and let the probability of interest q be the sum of the variables in Φ; i.e., $q = \sum_{j\in\Phi} \theta_j$. Suppose the prior distribution of $\Theta = (\theta_1, \theta_2, \ldots, \theta_k)$ is a Dirichlet distribution $D_k(\alpha_1, \alpha_2, \ldots, \alpha_k; \alpha_{k+1})$. A straightforward application of the Bayesian approach uses the training data D to update the prior distribution to obtain the posterior distribution $f(\Theta \mid D)$, which is a Dirichlet distribution $D_k(\alpha_1 + y_1, \alpha_2 + y_2, \ldots, \alpha_k + y_k; \alpha_{k+1} + y_{k+1})$. Then the posterior distribution $f(q \mid D)$ is derived from $f(\Theta \mid D)$. By Lemmas 1 and 2, $f(q \mid D)$ is a Beta distribution:

$$f(q \mid D) = Beta\left(\sum_{j\in\Phi}(\alpha_j + y_j), \alpha + n - \sum_{j\in\Phi}(\alpha_j + y_j)\right) \tag{1}$$

However, by the properties of Dirichlet distributions, there exists an alternative and simpler way to compute $f(q \mid D)$. We can first derive the prior distribution for the probability of interest $f(q)$ from the prior distribution of $f(\Theta)$. Then we can convert the training data D into a new set of training data D' in terms of q by computing the sum of the observations of our interest in D. Now, we can use D' to update the prior distribution of $f(q)$ to obtain the posterior distribution $f(q \mid D')$.

Definition 2. Perfect aggregation (Iwasa, Levin, & Andreasen, 1987; Wong, 1998) holds for a distribution f if $f(q \mid D) = f(q \mid D')$.

Theorem 1. *Perfect aggregation holds for the Dirichlet distribution.*

Proof: Sketch. (Azaiez, 1993) By Lemmas 1 and 2. □

For example, suppose that we are interested in the probability of showing an odd number in throwing a die. Let θ_j be the probability that the die shows number j in a trial, and let y_j be the number of trials that the die shows j in n trials. Then the probability of interest can be represented as $q = \theta_1 + \theta_3 + \theta_5$. In the straightforward approach, we derive the distribution $f(q \mid D)$ from the data $\{y_1, y_2, \ldots, y_6\}$, while in the alternative approach, we can use $D' = \{n, y_1 + y_3 + y_5\}$ instead and will obtain the same result.

An implication of perfect aggregation is that with Dirichlet distributions, to estimate the posterior probability of a union of events, there is no need to estimate the probabilities of

individual events. This implication allows us to show that discretizing continuous variables for a naive Bayesian classifier can be effective in approximating a variety of probability distributions. Another implication is that when perfect aggregation holds, identifying the exact outcome of an observation in D will not be necessary. In this case, we are only concerned about whether the result of an observation is an outcome included in the event corresponding to the probability of interest. Thus, when a probability of interest is known before training, perfect aggregation can simplify the training effort in identifying the outcome of an observation in D. This implication allows us to derive a lazy discretization method for a naive Bayesian classifier.

Two properties are important for perfect aggregation to hold for Dirichlet distributions. First, the order of the variables in a random vector with a Dirichlet distribution can be arbitrary. Second, the relationships of conditional independence among the variables are strong (Wong, 1998). For other multivariate distributions defined on the simplex, such as *generalized Dirichlet* distributions and *Liouville* distributions (Wilks, 1962), the conditions for perfect aggregation to hold are usually extremely restricted. Therefore, the Dirichlet assumption is critical for the perfect aggregation property to hold.

3. Dirichlet distributions in naive Bayesian classifiers

A naive Bayesian classifier classifies a query vector $\mathbf{x}$ by selecting class c that maximizes the posterior probability

$$p(c \mid \mathbf{x}) \propto p(c) \prod_{x \in \mathbf{x}} p(x \mid c), \tag{2}$$

where x is a variable in $\mathbf{x}$. $p(x \mid c)$ is the *class-conditional density* of x given class c. Let Θ denote the vector whose elements are the parameters of the density of $p(x \mid c)$. In a Bayesian learning framework, we assume that Θ is an uncertain variable (Heckerman, 1998) and can be learned from a training data set. This estimation is at the heart of training in a naive Bayesian classifier.

3.1. Dirichlet assumption

Suppose x is a discrete variable with $k+1$ possible values. In principle the class label c of the data vector $\mathbf{x}$ dictates the probability of the value of x. Thus the appropriate p.d.f. is a multinomial distribution and its parameters are a set of probabilities $\{\theta_1, \theta_2, \ldots \theta_{k+1}\}$ such that for each possible value X_j, $p(x = X_j \mid c) = \theta_j$ and $\sum_j \theta_j = 1$. Now, let $\Theta \equiv (\theta_1, \theta_2, \ldots \theta_k)$. We choose a Dirichlet distribution with parameters $\alpha_1, \ldots, \alpha_{k+1}$ as the prior for Θ. Given a training data set, we can update $p(x = X_j \mid c)$ by its expected value:

$$\hat{p}(x = X_j \mid c) = \frac{\alpha_j + y_{cj}}{\alpha + n_c}, \tag{3}$$

where n_c is the number of the training examples belonging to class c and y_{cj} is the number of class c examples whose $x = X_j$. Since a Dirichlet distribution is conjugate to multinomial

sampling, after the training, the posterior distribution of Θ is still a Dirichlet, but with the updated parameters $\alpha_j + y_{cj}$ for all j. This property allows us to incrementally train a naive Bayesian classifier.

In practice, we usually choose the Jaynes prior (Almond, 1995) $\alpha_j = \alpha = 0$ for all j and have $\hat{p}(x \mid c) = \frac{y_{cj}}{n_c}$. However, when the training data set is too small, this often yields $\hat{p}(x \mid c) = 0$ and impedes the classification. To avoid this problem, another popular choice is $\alpha_j = 1$ for all j. This is known as *smoothing* or *Laplace's estimate* (Cestnik & Bratko, 1991).

If x is a continuous variable, a conventional approach is to assume that $p(x \mid c) = N(x; \mu_c, \sigma_c^2)$, where $N(x; \mu_c, \sigma_c^2)$ is the p.d.f. of a normal distribution. In this case, training involves learning the parameters μ_c and σ_c from the training data (Duda & Hart, 1973). This approach has been shown to be less effective than discretization when $p(x \mid c)$ is not normal (John & Langley, 1995), and discretization is often used. Generally, discretization involves partitioning the domain of x into $k + 1$ intervals as a pre-processing step. Then we can treat x as a discrete variable with $k + 1$ possible values and conduct the training and classification.

More precisely, let I_j be the j-th discretized interval. Training and classification in a naive Bayesian classifier with discretization uses $\hat{p}(x \in I_j \mid c)$ as an estimate of $\hat{p}(x \mid c)$ in Eq. (3) for each continuous variable. This is equivalent to assuming that after discretization, the class-conditional density of x has a Dirichlet prior. We call this assumption "*Dirichlet discretization assumption*." Apparently, this assumption is adopted implicitly by all well-known discretization methods, including ten-bin, entropy-based, etc. See Section 4.1 for a brief survey.

3.2. Partition independence assumption

The Dirichlet discretization assumption is reasonable because of another implicit assumption described below. Let $f(x \mid c)$ be the "true" probability density function of $p(x \mid c)$. Assuming that $f(x \mid c)$ is integrable everywhere. Then for any discretized interval I_j, the "true" probability of $p(x \in I_j \mid c) = \int_{I_j} f(x \mid c)\, dx$. By choosing equivalent sample size α, the Dirichlet parameter corresponding to random variable "$x \in I_j \mid c$" is $\alpha \int_{I_j} f(x \mid c)\, dx$. We call this assumption "*partition independence assumption*." By the partition independence assumption, any discretization of x can have a Dirichlet prior.

The partition independence assumption implies that for any interval, the Dirichlet parameter corresponding to this interval depends only on the area below the curve of the p.d.f. $f(x \mid c)$, but is independent of the shape of the curve in the interval. In figure 1, the shape of the p.d.f. curves in $[a_1, a_2]$ are different, yet the Dirichlet parameters corresponding to this interval for these two p.d.f's are identical.

4. Why discretization works for naive Bayesian classifier

Based on the properties of Dirichlet distributions, this section explains the empirical findings presented in previous work concerning the performance of the discretization methods for naive Bayesian classifiers (Dougherty, Kohavi, & Sahami, 1995; Kohavi & Sahami, 1996;

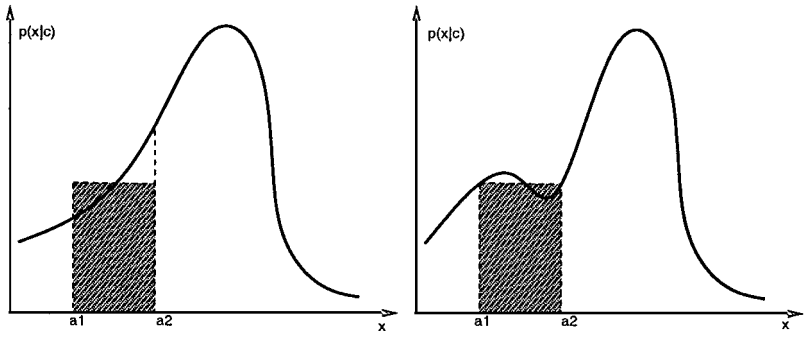

Figure 1. Partition independence assumption.

Domingos & Pazzani, 1997) and extend the explanation to conclude that a wide variety of discretization methods all perform similarly.

4.1. Review of the well-known discretization methods

This section reviews four well-known discretization methods. A more comprehensive review can be found in Dougherty, Kohavi, and Sahami (1995).

4.1.1. Equal width interval binning. Equal width interval binning involves sorting the observed values of a continuous variable and dividing the range of observed values for the variable into k equally sized bins, where k is usually set to ten.

4.1.2. Bin-logl. This is an equal width interval discretization method with the number of bins $k = \max\{1, 2\log(l)\}$, where l is the number of distinct observed values for each attribute. The heuristic was chosen based on examining S-plus's histogram binning algorithm (Spector, 1990).

4.1.3. Holte's 1R discretizer. This method, referred to here as 1RD (One-Rule Discretizer) (Holte, 1993), sorts the observed values of a continuous variable and attempts to greedily divide the domain of the feature into bins where each contains only instances of one particular class. Since such a scheme could possibly lead to one bin for each observed real value, the algorithm is constrained to forms bins of at least some minimum size (except the rightmost bin). Holte suggests a minimum bin size of six based on an empirical analysis.

4.1.4. Recursive minimal entropy partitioning. This method, referred to here as entropy-based method, is based on a minimal entropy heuristic developed by Fayyad and Irani (1993). This method uses the class information entropy of candidate partitions to select bin boundaries for discretization. If we are given a set of instances S, a variable x, and a partition boundary T. Then T partition the set S of examples into the subsets S_1 and S_2. Let there be k classes $C_1, \ldots, C_k$ and let $\Pr(C_i, S)$ be the proportion of examples in S that

have class C_i. The *class entropy* of a subset S is defined as:

$$Ent(S) = -\sum_{i=1}^{k} P(C_i, S)\log(P(C_i, S)). \tag{4}$$

Then the class information entropy of the partition induced by T, denoted $E(x, T; S)$ is given by:

$$E(x, T; S) = \frac{|S_1|}{|S|}Ent(S_1) + \frac{|S_2|}{|S|}Ent(S_2). \tag{5}$$

For a given variable x, the boundary $T_{\min}$ which minimizes Eq. (5) over all possible partition, a boundary is selected as a binary discretization boundary. This method can then be applied recursively to both of the partitions induced by $T_{\min}$ until some stopping condition is achieved, thus creating multiple intervals on the feature x.

Fayyad and Irani make use of the *Minimal Description Length Principle* to determine a stopping criterion for their recursive discretization method. Recursive partitioning within a set of values S stops iff

$$Gain(x, T; S) < \frac{\log_2(N-1)}{N} + \frac{\triangle(x, T; S)}{N}, \tag{6}$$

where N is the number of instances in the set S,

$$Gain(x, T; S) = Ent(S) - E(A, T; S), \tag{7}$$

$$\triangle(x, T; S) = \log_2(3^k - 2) - [kEnt(S) - k_1Ent(S_1) - k_2Ent(S_2)], \tag{8}$$

and k_i is the number of class labels represented in the set S_i. Since the partitions along each branch of the recursive discretization are evaluated independently using this criteria, some areas in the continuous spaces will be partitioned very finely whereas others will be partitioned coarsely.

4.2. Why discretization outperforms parameter estimation

Previously, Dougherty, Kohavi, and Sahami (1995) empirically verified that a variety of well-known discretization methods in handling continuous variables could perform similar to or better than the methods that assumed normal priors for the continuous variables in many situations. In addition, those discretization methods generally achieve higher accuracy than the methods with normal priors for the data sets in which the continuous variables are not actually normally distributed. An enhanced approach of the method with normal priors proposed by John and Langley (1995) achieves only limited performance improvement for some data sets.

We conducted a simple experiment to see if we can draw the same conclusion. This experiment uses two data sets `iris` and `glass` from UCI ML repository (Blake & Merz, 1998). All continuous variables in data set `iris` are known to fit normal distributions well,

while data set glass is too small to determine appropriate probability distributions for its continuous variables. We applied a genetic algorithm to search for the best distributions and their corresponding parameters for the variables in the two data sets. The genetic algorithm quickly found a set of normal distributions for set iris and achieved a $95.00 \pm 2.52\%$ cross-validation accuracy. On the contrary, it took about three weeks for the genetic algorithm to search a distribution for glass, and the best accuracy is only $61.50 \pm 3.96\%$, worse than the accuracy achieved by most discretization methods. This again confirms that when the distributions of continuous variables cannot be modeled as normally distributed, discretization is generally a better approach than assuming that the variables are normally distributed.

When the density function of a continuous variable x is known, the conditional probability $p(x \mid c)$ will be used in Eq. (2) for classification. In a discretization approach, $p(x \mid c)$ is replaced by the estimate $\hat{p}(x \in I_j \mid c)$ that represents the probability that x is in interval I_j when the class is c. This estimate is then used in Eq. (2) for classification. Based on the Dirichlet assumption, the random vector of the estimates $\hat{p}(x \in I_j \mid c)$ has a Dirichlet distribution.

Let $f(x \mid c)$ be the conditional p.d.f. of x. When $f(x \mid c)$ is not integrable everywhere, we have

$$p(x \in I_j \mid c) = \int_{x \in I_j} f(x \mid c)\, dx \approx \sum_{\Delta \subseteq I_j} \hat{p}(x \in \Delta \mid c),$$

where Δ is a disjoint sub-interval in I_j. The precision in estimating $\hat{p}(x \in I_j \mid c)$ can be raised by increasing the number of sub-intervals in I_j to make $f(x \mid c)$ integrable everywhere in each sub-interval Δ. Therefore, $\hat{p}(x \in I_j \mid c)$ can be estimated arbitrarily close to the real $p(x \in I_j \mid c)$ regardless of the original density function $f(x \mid c)$.

Since perfect aggregation always holds for the Dirichlet distribution (Theorem 1), by the Dirichlet assumption, deriving $\hat{p}(x \in I_j \mid c)$ can be achieved by estimating the $p(x \in \Delta \mid c)$ such that each disjoint sub-interval in Δ is integrable everywhere. By the partition independence assumption, the estimate of $\hat{p}(x \in I_j \mid c)$ is independent of the shape of $f(x \mid c)$ in interval I_j when $\int_{x \in I_j} f(x \mid c)dx$ is fixed. Therefore, discretization methods can construct a more appropriate prior for a continuous variable that is unlikely to have a normal distribution. This implies that discretization methods are more widely applicable than the methods by assuming normal priors for continuous variables.

When the training data set is not sufficiently large, discretization may not yield accurate estimations. Parameter estimation, however, suffers the same problem. How data sizes affect the performance of discretization methods will be discussed in Section 4.3.

4.3. Why different discretization methods achieve similar performance

Dougherty, Kohavi, and Sahami (1995) conducted a comparative study of a variety of discretization methods and concluded that an entropy-based method (Fayyad & Irani, 1993) performs slightly better in some data sets but on average all discretization methods performs approximately the same. Subsequent work also confirmed their finding.

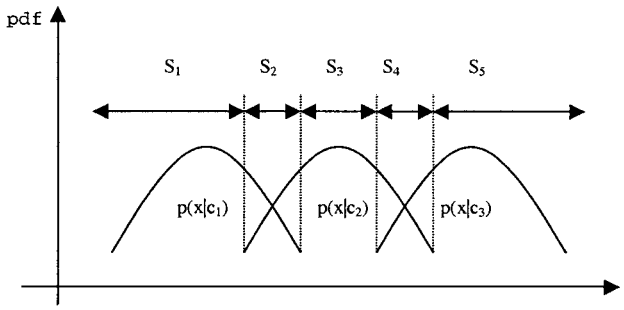

Figure 2. Conditional distributions of the synthesized data.

Suppose we have the "true" densities of $p(x \mid c)$, it can be shown that Eq. (2) (i.e., the Bayes rate) is the optimal classification accuracy achievable by any classifier (Duda & Hart, 1973). With discretization, to achieve the optimal classification accuracy, $\hat{p}(x \in I \mid c)$ must simulate the role of $p(x \mid c)$ by distinguishing the class that gives x high density from the class that gives x low density. More precisely, if $p(x = X_j \mid c_i)$ is the largest among all classes c_i, then at least we should require that $\hat{p}(x \in I \mid c_i)$ is the largest among all classes, too. Otherwise, performance degradation may still occur.

Consider a simple case with only one continuous variable x with uniformly distributed classes. We can plot the curves of the conditional densities as the one shown in figure 2. The domain of x can be divided into regions by *decision boundaries*, which correspond to the intersection points of the curves where ties occur among the largest conditional densities. The optimal classification is to pick the class c such that $p(x \mid c)$ is the largest, and the pick of the class is different when x is on different sides of a decision boundary.

Suppose a discretization method creates an interval I that contain a decision boundary. For x values in the interval but at the different sides of the decision boundary, the classifier should pick a different class for each value. But we will have the same $\hat{p}(x \in I \mid c)$ for these x values. That is, x values at one of the two sides of the decision boundary will be misclassified. Therefore, to minimize the distortion, the boundaries of intervals that contain decision boundaries should be close to the decision boundaries, or these intervals should be as narrow as possible.

On the other hand, the widths of intervals that contain no decision boundary can be as large as possible. Their width do not affect too much on the classification performance but when these intervals are too narrow, the number of examples in the training data set in the intervals will be too small to allow accurate estimation of $\hat{p}(x \in I \mid c)$. In this case, discretization will cause performance degradation. A loose bound can be derived as follows. Let n be the size of the training data set. The number of examples that will fall in an interval I has a binomial distribution with parameters n and $p(x \in I \mid c)$ and has expected value $n * p(x \in I \mid c)$. If we want the estimation to be accurate to two digits, we will need $n > 100/\hat{p}(x \in I \mid c)$. This bound is loose but is sufficient to illustrate our idea. The entropy-based method and 1RD are designed to avoid narrow intervals, and thus their performance may be slightly better in some domains.

In summary, to avoid performance degradation, a discretization method should partition the domain of a continuous variable into intervals such that their cut points are close to decision boundaries to minimize the distortion due to the discretization. The selection of the cut points at the intervals that contain no decision boundaries has almost no impact on the classification performance, but their width should be large enough to cover sufficient examples.

The above analysis can be extended to multivariate cases and the "right" choice of cut points can be widely spread. Since a naive Bayesian classifier takes all variables into account simultaneously, the impact of a "wrong" discretization for one variable can be absorbed by other variables under zero-one loss performance measure. Also, due to the way a naive Bayesian classifier updates its posterior probabilities, correct classification of a given example depends only on the interval containing the values of that example, and is completely independent of how other regions are discretized. As a result, a wide variety of discretization methods can have similar performance in general.

5. Empirical evidence of the analysis of discretization

In this section, we report the empirical verification of our explanation using synthesis and real data sets.

5.1. Controlled experiments

We synthesized an artificial data set composed of 1500 examples with three different classes. Each class has the same number of examples. Each example has two continuous attributes A_1, A_2 and a class label. The values of A_1 and A_2 are generated by normal distributions $\{N(10, 2), N(15, 2)\}$ for class 1, $\{N(15, 2), N(20, 2)\}$ for class 2, and $\{N(20, 2), N(10, 2)\}$ for class 3. For each attribute, we divided its domain into 5 intervals as shown in figure 2 such that S_2 and S_4 contain the decision boundaries for each variable.

In the first experiment, we investigated how the distance of the cut points of S_2 and S_4 to the intersection points affect the performance. S_2 and S_4 are initialized with a small width and their boundaries are extended with a constant percentage of the initial width of S_3. The decision boundaries within S_2 and S_4 were kept in the middle such that the regions at both sides of the decision boundaries always have the same width. For each resulting discretization, we ran a five-fold cross-validation to obtain an average accuracy. Figure 3 plots the results. As we predicted, when S_2 and S_4 become wide, the accuracies drop rapidly.

The second experiment is to see if there are other factors that may cause significant changes of the accuracies in addition to the width of S_2 and S_4. We first kept the boundaries of S_2 and S_4 unchanged (their widths were set as in ten-bin) and randomly partitioned S_1, S_3, and S_5 into at most fifty intervals. We repeated the random partition thirty times and ran a five-fold cross-validation for each repetition. The variance of the resulting accuracies is 0.27. The average accuracy is 96.25% with the maximum and minimum accuracies being 96.67% and 95.47%, respectively. We repeated the same experiment, but this time, the entire domains are randomly partitioned. By contrast, the variance of the resulting accuracies is 3.84, which is 14 times higher than the result of the controlled random partition. The average

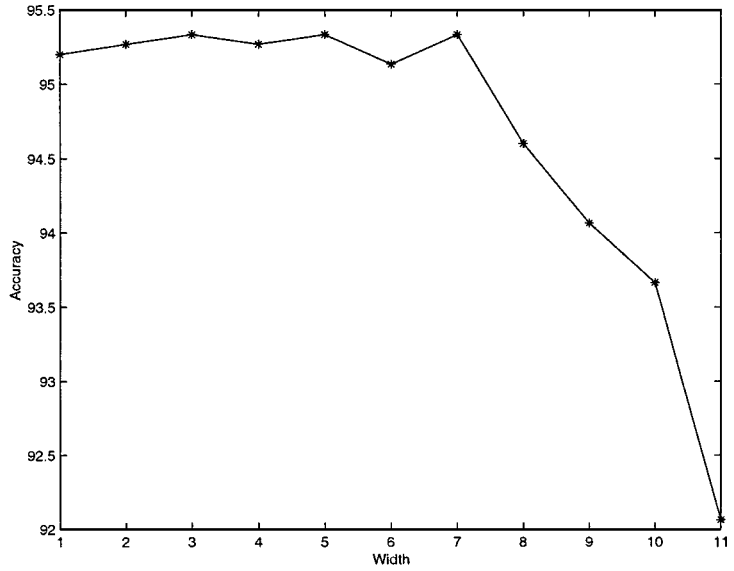

Figure 3. Accuracies drop when we increase the widths of the intervals that contain the intersection points.

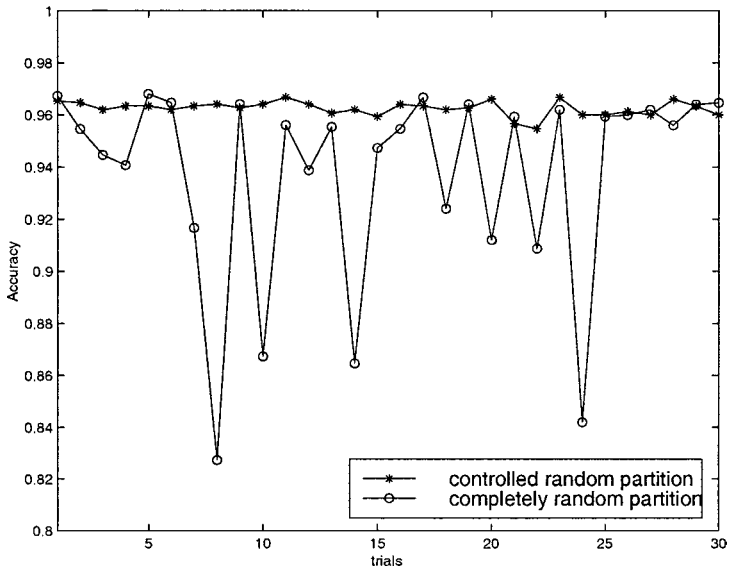

Figure 4. Accuracies of controlled random partition and complete random partition.

accuracy is 93.79% with the maximum and minimum accuracies being 96.80% and 82.73%, respectively. Figure 4 plots the detailed accuracy data of each individual trial. The results show that when the widths of S_2 and S_4 are fixed, the partitions of the other intervals have only tiny impact on the classification accuracy.

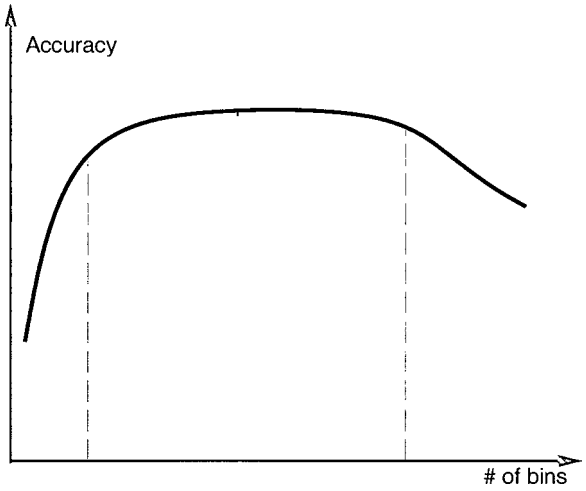

Figure 5. Predicted curve pattern of accuracy vs. number of bins.

The third experiment is to show that "equal-width bins" such as ten-bin can be effective in many cases. In fact, if the number of bins is not too "extreme," the performance will not be significantly different. Our analysis predicts that when the number of intervals is two, it is likely that these intervals will contain decision boundaries and the performance will be generally poor. As the number of bins increases, the accuracies will improve and reach a plateau. When the number of bins is extremely large, the widths of the bins will be too small for all intervals and the accuracies will drop gradually. That is, the curve of the accuracy as a function of the number of bins will have a characteristic shape like an up-side-down "J" as shown in figure 5. Moreover, how fast the curve drops will depend on the size of the training data set. A small data set will cause the accuracy to drop faster.

We conducted the experiment on two data sets: our synthesized data set (with 1500 examples) and its subset (with 300 examples). The resulting curves are given in figure 6, which confirms our prediction.

5.2. Experiments on real data sets

The next question is whether we can obtain the same results with real data sets. Table 1 lists the sixteen real data sets selected from UCI ML repository (Blake & Merz, 1998) for our experiments. Ten of them are "pure" continuous data sets whose attributes are all numeric, and six of them have both types of attributes.[1] The first experiment is similar to the third experiment described in 4.1. In this experiment, we partitioned each continuous variable into two equal-width bins at first, and then increased the number of the equal-width bins by one until 50. All accuracy figures are obtained by using five-fold cross-validations.

Figure 7 shows the results for the "pure" continuous data sets. As expected, the curves for most data sets match the pattern of the curve given in figure 5, especially for large data sets. The variances of the accuracy for most data sets are found to be surprisingly small.

Table 1. Data sets for our experiments. The fourth column shows the accuracies of a naive Bayesian classifier with the continuous variables removed, and the fifth column shows the accuracies with all variables.

Data set (classes)	Variable. cont:disc	Data size	NB acc% w/o cont.	NB acc% w cont.
Australian (2)	6:8	898	85.80 ± 3.06	85.22 ± 2.73
Breast-W (2)	30:0	569	–	–
Cleve (2)	6:7	303	81.24 ± 3.69	81.86 ± 5.83
CRX (2)	6:9	690	83.77 ± 3.93	84.93 ± 2.07
German (2)	24:0	1000	–	–
Glass (7)	9:0	214	–	–
Heart (2)	5:8	270	85.19 ± 3.41	83.33 ± 1.17
Hepatitis (2)	6:13	155	83.87 ± 3.53	87.10 ± 4.56
Hypothyroid (2)	7:18	3190	95.23 ± 0.73	97.12 ± 0.56
Iris (3)	4:0	150	–	–
Liver (2)	6:0	345	–	–
PIMA (2)	8:0	768	–	–
Vehicle (4)	18:0	846	–	–
Waveform (3)	21:0	5000	–	–
Wine (3)	13:0	178	–	–
Yeast (10)	8:0	1484	–	–

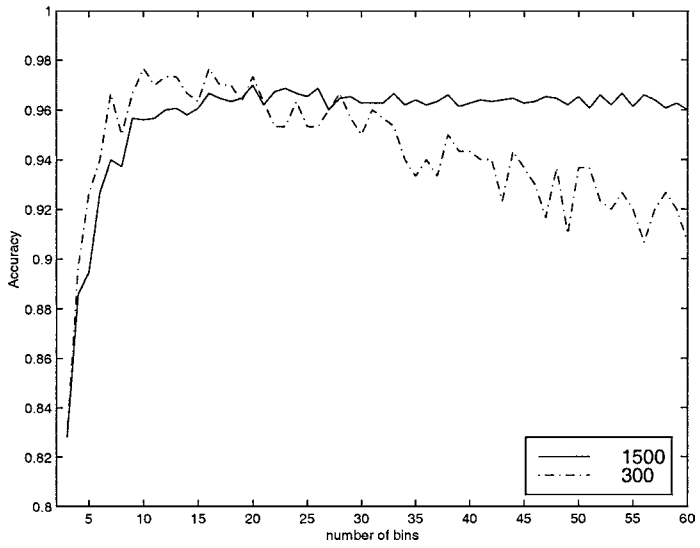

Figure 6. Experimental results match our prediction. Also, with fewer data, accuracies drop faster as the number of bins increases.

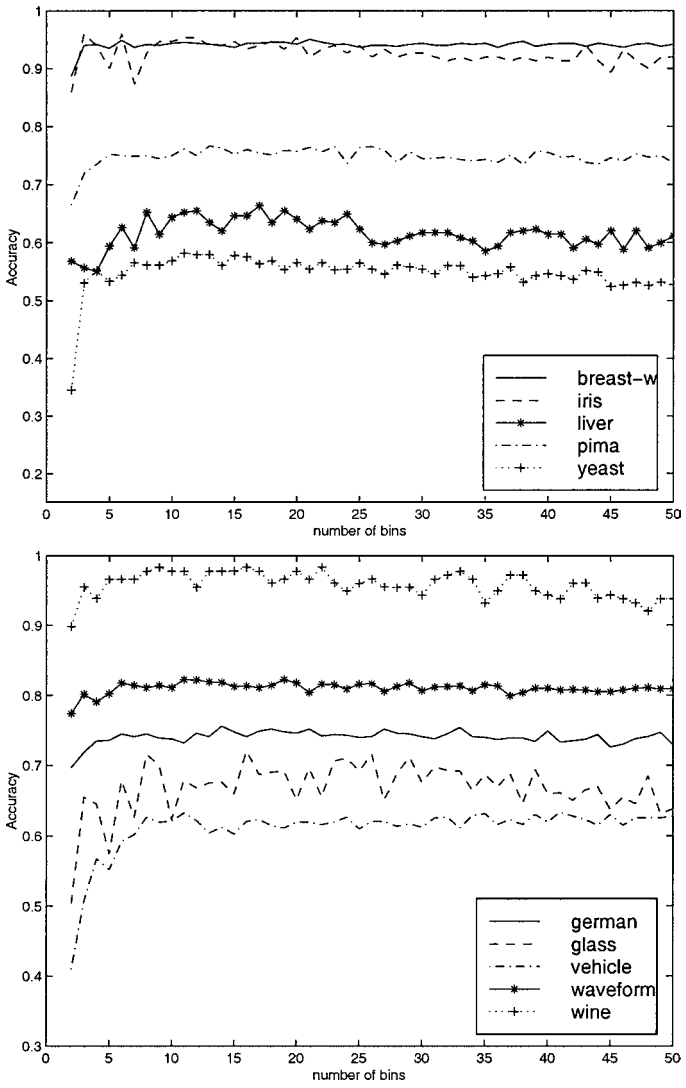

Figure 7. Accuracy results by increasing the number of equal-width bins for the "pure" continuous data sets.

As long as the numbers of equal-width bins are not too large or too small, the accuracies are approximately the same. The curves for some data sets (`glass`, `iris`, `liver`, `wine`) are rougher than others, since the variances of the accuracy appear to depend on the size of the data sets. In general, small data sets (with less than 300 examples) have relatively larger variances. As discussed earlier, this is because the examples are not sufficient for accurate probability estimation. Data set `glass` has a particularly zigzag curve because the number of examples in this data set is small. In addition, this data set has seven classes, which produce too many decision boundaries and the accuracies become sensitive to discretization.

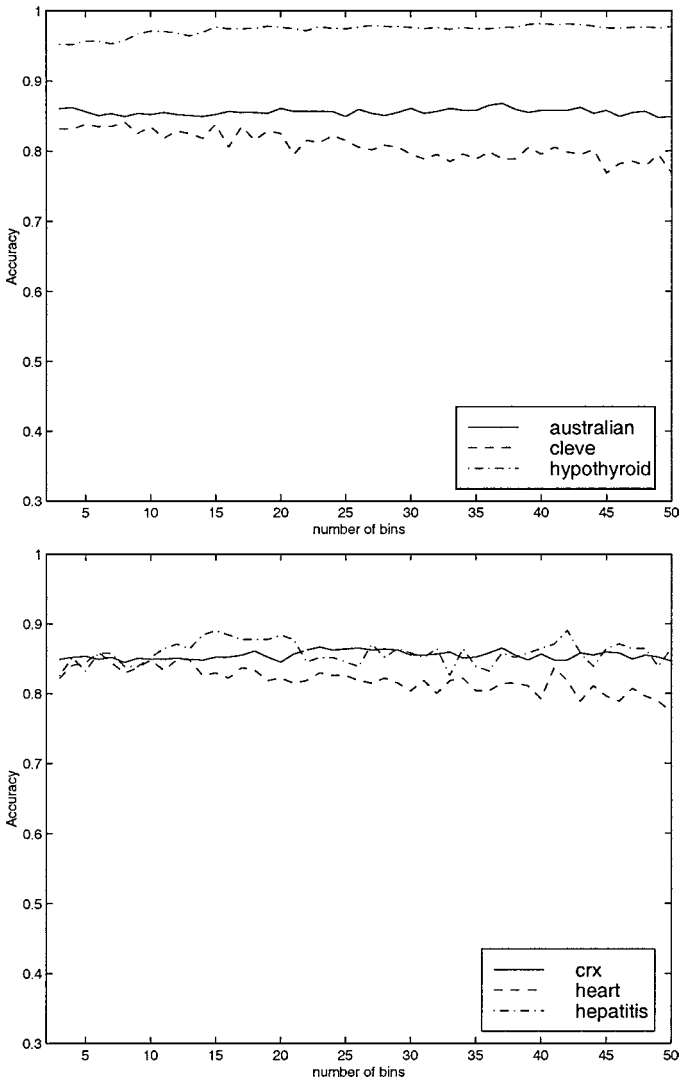

Figure 8. Accuracy results by increasing the number of equal-width bins for the "hybrid" data sets.

In the above experiment, we focus on domains that are entirely continuous. We also performed the same experiment for some data sets with hybrid domains. Figure 8 shows the results for these "hybrid" data sets. The curves of those hybrid data sets again generally match the pattern but flatten out at the beginning instead of growing. We examined those data sets by comparing the accuracies of a naive Bayesian classifier with and without their continuous variables. Table 1 gives the resulting accuracy figures. The fourth column shows the accuracies of a naive Bayesian classifier with the continuous variables removed, and the fifth column shows the accuracies with all variables.[2] Since the resulting pairs of

the accuracy figures are about the same, the continuous variables in those data sets have almost no impact in classifying test data. When a continuous variable in a data set is nearly dormant in classification, the distributions of different classes of that variable will be similar everywhere. Therefore, a naive Bayesian classifier with only two bins for those data sets can still achieve a high accuracy.

We also conducted a similar experiment for all data sets to visualize the effect when the number of bins is very large. Basically, we partitioned each continuous variable into two equal-width bins at first and increased the number of bins by $\lfloor 5\% * n \rfloor$ in each iteration, where n is the training data size. For data set `waveform`, we increased the number of bins by fifty for each iteration because its size is too large. Figures 9 and 10 plot the resulting curves for the "pure" continuous data sets and the "hybrid" data sets respectively. The index of the horizontal axis indicates how many percentages of n are added to the number of equal-width bins. As expected, the curves suggest a decreasing trend, minor in most cases, as the number of bins increases.

The results presented above does not exclude the possibility that by carefully selecting the cut points, the performance will be improved drastically. Therefore, we conducted a similar experiment with the entropy-based method (Fayyad & Irani, 1993). Since the entropy-based method always selects a cut point from training data at the boundary where the classes switch, in theory it is more likely for this method to select a cut point near a decision boundary. For each data set, we recursively called the entropy heuristic to select cut points and reported the five-fold cross-validation accuracy. In each recursion, cut points are selected in the resulting intervals of the previous calls without applying MDL stopping criterion (Fayyad & Irani, 1993). Figures 11 and 12 show the resulting curves. The horizontal axis indicates how many levels of the recursion have been invoked. At the lth level of the recursion, a continuous variable will have at most 2^l cut points. However, when all of the data contained in an interval have the same class, no cut point will be selected. Therefore, the number of intervals is generally considerably less than 2^l, and each variable may be partitioned into a different number of intervals.

The resulting curves show that the accuracies with different levels of recursions are surprisingly similar for all data sets. Their shapes match the pattern of the curves in figure 5. This again confirms that discretization methods can perform similarly even though they select radically different sets of cut points.

6. Lazy discretization

Our "lazy" discretization method is derived from the properties of Dirichlet distributions. If the Dirichlet assumption is appropriate, the lazy method should perform as well as other "eager" discretization methods, including the well-known ones. Therefore, a motivation to introduce this method is to justify the selection of Dirichlet priors and verify our analysis given in previous sections. Another motivation is that the lazy method can be extended to classify set and interval data, which will be described in Section 7.

The lazy discretization method waits until one or more test data are given to determine the cut points for each continuous variable. This method produces only a pair of cut points surrounding the value of each variable. That is, it creates only one interval (denoted as I)

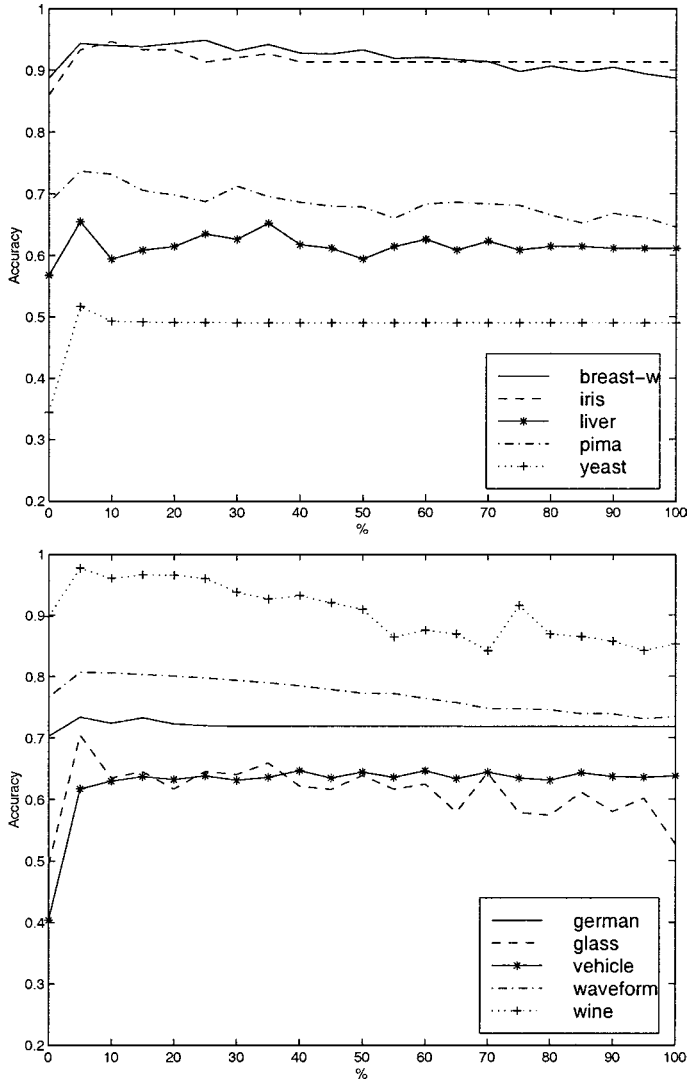

Figure 9. Accuracy results by increasing the number of equal-width bins with a constant percentage of data size for the "pure" continuous data sets.

for each variable and leaves the other region untouched. From the training data, we can estimate $\hat{p}(x \in I \mid c)$ by Eq. (3) and use this estimate to classify the query vector.

6.1. Derivation of the lazy discretization method

The lazy discretization method is derived from perfect aggregation and other properties of Dirichlet distributions. Let x be a continuous variable in a naive Bayesian classifier

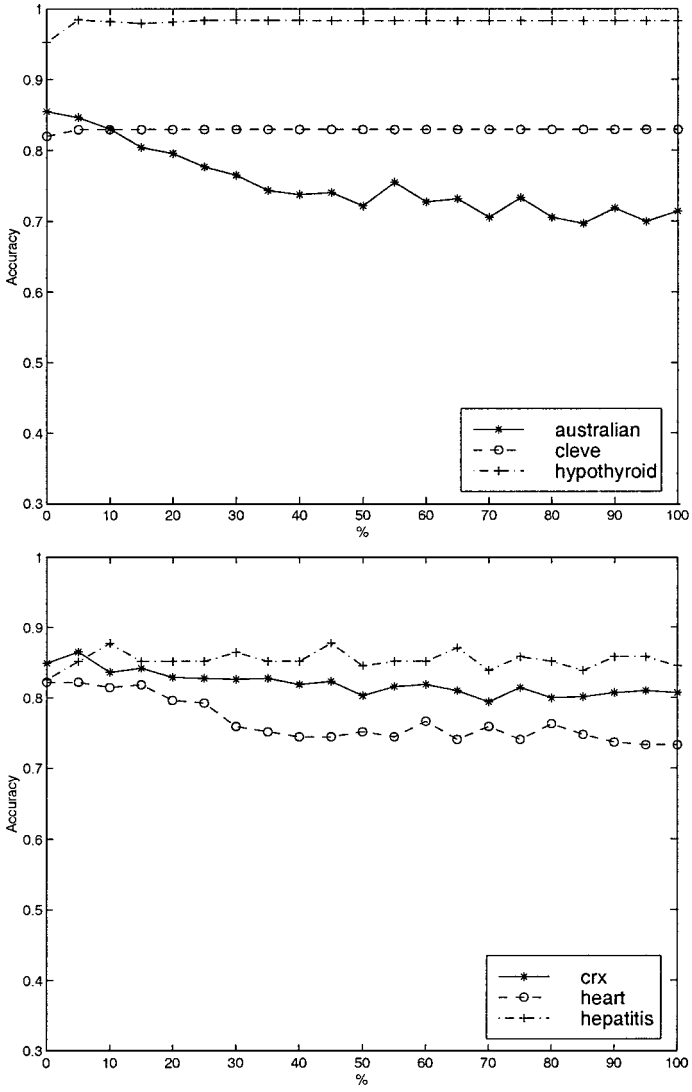

Figure 10. Accuracy results by increasing the number of equal-width bins with a constant percentage of data size for the "hybrid" data sets.

and its value is X for a given query vector. The lazy discretization method will create an interval $I = [X - 0.5\delta, X + 0.5\delta]$ for x, where δ is a constant. If the partition independence assumption holds, by Lemma 2, "$x \mid c$" will have a Beta prior with parameters $\alpha \int_I p(x \mid c)dx$ and $\alpha(1 - \int_I p(x \mid c)dx)$, where α is an equivalent sample size. By perfect aggregation, we can estimate $\hat{p}(x \in I \mid c)$ by counting how many c examples with $x \in I$ and how many are not. In other words, there is no need to check the exact value of x for those examples whose value of x is not in I.

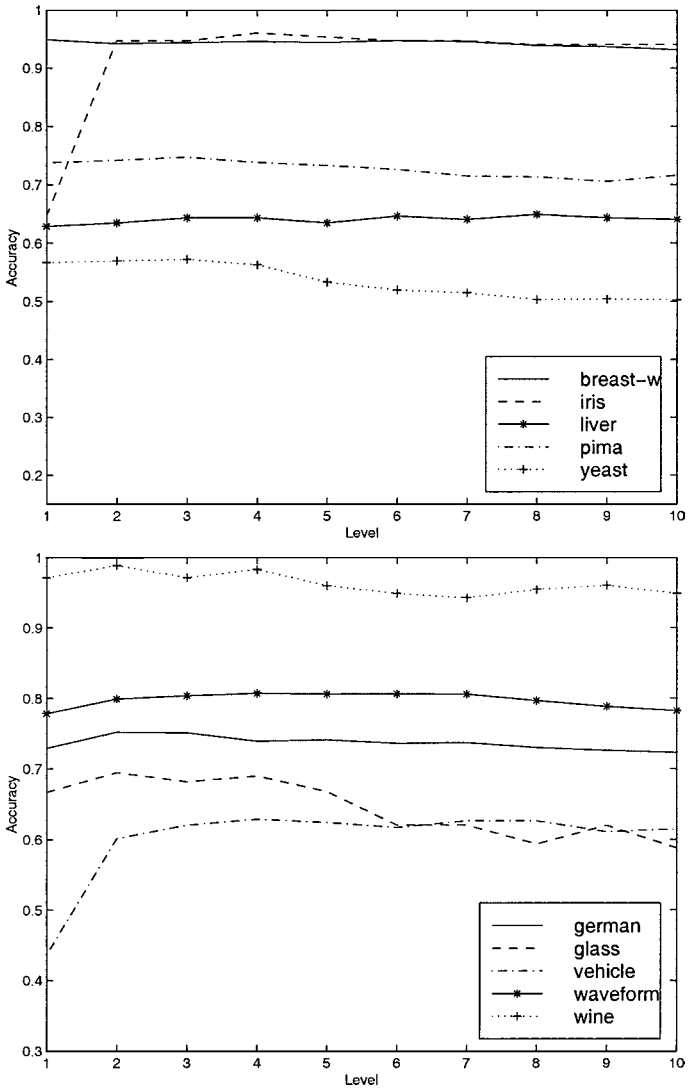

Figure 11. Accuracy results by using entropy to determine cut points for the "pure" continuous data sets.

More precisely, given a data set of training examples and a query vector to be classified, the lazy discretization method works as follows:

1. for each continuous variable $x = X$, determine cut points $X - 0.5\delta$, $X + 0.5\delta$ surround data value;
2. treat that variable as if it is a discrete variable with only two possible outcomes;
3. train the naive Bayesian classifier with the training examples;
4. classify the query vector.

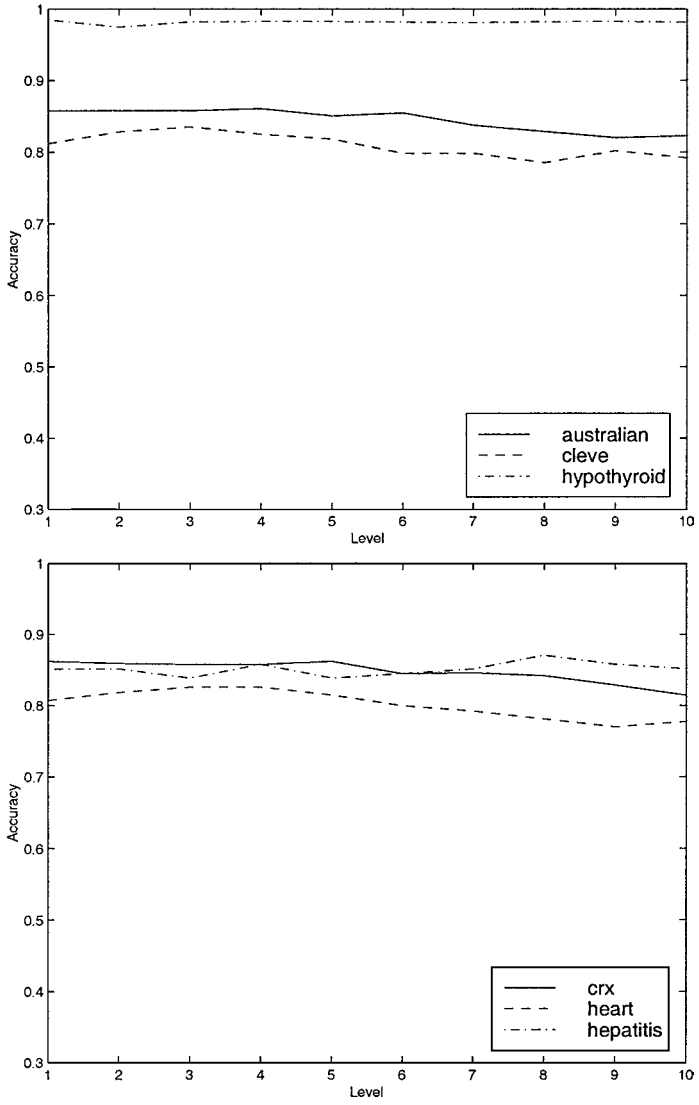

Figure 12. Accuracy results by using entropy to determine cut points for the "hybrid" data sets.

There can be different instantiations of this method, depending on how we determine the width of the interval δ. For example, we can select δ to be equal to the width of the intervals created by ten-bin. We call this instantiation of the lazy discretization method as "lazy ten-bin." Similarly, we can have "lazy bin-log" by assigning δ to be half of the width of $\log l$ equal bins and "lazy entropy" by assigning cut points surrounding x according to minimum entropy heuristics.

An advantage of this method is that when the test data contain interval values, we can simply assign I to be the given interval and no more discretization is necessary. This point will be elaborated in Section 7.

6.2. *Fast lazy discretization*

To speed up the estimation of $\hat{p}(x \in I \mid c)$, we can divide the domain of each continuous variable into a large number of equal-width bins in advance, then count the number of the training examples in each bin for each class c and cache the results in a hash table. During the classification time, an approximate value of $\hat{p}(x \in I \mid c)$ can be computed by table lookup. The larger the number of bins that we divide in advance, the closer to the real values the estimated probabilities will be. In our experiments, we divided each continuous variable into one thousand equal-width bins and cache the example counts in advance. We call this variation of the lazy discretization method as "fast-lazy discretization."

6.3. *Experimental results*

We empirically compared lazy ten-bin, fast-lazy ten-bin, ten-bin, entropy-based, bin-log l and a normal version of a naive Bayesian classifier on data sets from UCI ML repository. We ran a five-fold cross-validation for each data set and reported the average and the standard deviation of the accuracies. Tables 2 and 3 give the results. The best accuracy for each data set is indicated in boldface.

Table 2 also compares the elapsed time of one training-test trial of cross-validation of lazy ten-bin and fast-lazy ten-bin. Obviously, the elapsed time of fast-lazy ten-bin is much less than that of lazy ten-bin.

We also use ANOVA with the Bonferroni procedure for multiple comparisons statistics (McClave & Dietrich, 1991) to compare the lazy method with the well-known methods. The *Required Difference* (RD) value for each data set is given in Table 4. The difference between any two accuracies of a data set in Tables 2 and 3 must be at least as large as the corresponding *RD* value in order to be considered statistically significant at the 90% confidence level for the experiment as a whole.

The results show that the performance of fast-lazy ten-bin is approximately the same as lazy ten-bin and both perform as well as well-known discretization methods. Among all six discretization methods (lazy ten-bin, fast-lazy ten-bin, ten-bin, entropy-based and bin-log l), no method is statistically significantly better or worse than the others for all data sets. Also, no discretization method is significantly worse than normal in all data sets, but normal is significantly worse than all discretization methods in six data sets (`australian`, `crx`, `glass`, `liver`, `vehicle` and `yeast`) as indicated with a superscript "$-$" sign in Table 3.

7. Classifying set and interval data

This section presents an extension of lazy discretization method that allows a naive Bayesian classifier to classify a query vector containing set or interval values.

Table 2. Performance comparison of lazy and fast-lazy discretization methods in terms of the classification accuracy and the computational cost.

	Accuracy (%)		Elapsed time (sec)	
Data set	Lazy	Fast-lazy	Lazy	Fast-lazy
Australian	**85.94 ± 2.85**	85.36 ± 2.95	277	2
Breast-W	**96.00 ± 3.08**	94.39 ± 2.95	1081	4
Cleve	83.60 ± 4.19	82.83 ± 1.83	137	<1
CRX	85.36 ± 2.01	85.94 ± 2.73	314	2
German	74.00 ± 2.98	**75.30 ± 1.42**	3465	10
Glass	64.43 ± 5.29	63.48 ± 6.77	159	<1
Heart	83.33 ± 2.03	82.29 ± 3.17	111	<1
Hepatitis	**87.74 ± 4.16**	85.00 ± 6.19	18	<1
Hypothyroid	97.88 ± 0.38	96.82 ± 0.60	11253	29
Iris	96.00 ± 2.49	**97.00 ± 4.33**	10	<1
Liver	64.64 ± 3.26	63.58 ± 3.17	105	<1
PIMA	**76.18 ± 4.84**	76.12 ± 3.51	455	1
Vehicle	61.23 ± 1.75	61.08 ± 1.71	3337	6
Waveform	81.75 ± 3.22	81.41 ± 3.14	75896	43
Wine	97.19 ± 1.81	97.78 ± 1.67	50	<1
Yeast	57.53 ± 3.91	**58.72 ± 4.26**	2588	4
Average	80.80	80.44	6203.5	6.75

7.1. Training and classification for set and interval data

In previous work of naive Bayesian classifiers, variables usually have *point values* in the sense that their values are atomic. In fact, a variable may have a *set value* if its value is a set, or have an *interval value* if its value is an interval. If a set-valued variable whose value contains interval members, then we call this variable a *multi-interval* variable. For example, variable $x_1 = \{red,\ yellow,\ green\}$ has a set value, $x_2 = [20.5, 38.5]$ has an interval value, and $x_3 = \{[20.5, 38.5], [49.6, 60.4]\}$ has a multi-interval value.

In many situations, it is more appropriate to model a variable as set-valued or interval-valued than as point-valued. One of such situations is when the variable represents a time frame within a certain period of time in temporal reasoning. The other example is when each query vector represents the observation of a collection instead of an individual. Interval values are also more appropriate when the data is obtained from a sensor that returns an interval instead of a point value. The interval can be formed by the maximum and minimum values detected by the sensor.

To classify a query vector that contains an interval-valued variable, no discretization is necessary because we can simply estimate the probability of the interval from training data set, as if the interval were a discretized interval in lazy discretization. We can also classify set-valued and multi-interval valued data in a similar manner. Let x be the variable that has

Table 3. Accuracies of naive Bayesian classifier using three well-known discretization methods and using parameter estimation assuming normal.

Data set	Ten bins	Entropy	Bin-log l	Normal
Australian	85.22 ± 2.73	$\mathbf{85.94 \pm 0.98}$	84.78 ± 1.94	$77.10 \pm 2.27^{-}$
Breast-W	94.37 ± 2.16	94.37 ± 1.93	94.37 ± 1.84	92.79 ± 2.61
Cleve	81.86 ± 5.83	$\mathbf{83.83 \pm 3.23}$	80.86 ± 3.87	82.86 ± 5.14
CRX	84.93 ± 2.07	$\mathbf{86.09 \pm 1.97}$	83.91 ± 2.03	$77.83 \pm 2.13^{-}$
German	73.80 ± 2.77	74.70 ± 2.84	72.40 ± 3.14	70.00 ± 2.49
Glass	62.34 ± 6.03	$\mathbf{67.12 \pm 3.60}$	66.89 ± 4.02	$47.58 \pm 4.96^{-}$
Heart	83.33 ± 1.17	82.59 ± 2.51	80.74 ± 4.16	$\mathbf{84.44 \pm 2.22}$
Hepatitis	87.10 ± 4.56	85.81 ± 5.28	85.81 ± 5.24	84.51 ± 4.74
Hypothyroid	97.12 ± 0.56	$\mathbf{98.29 \pm 0.52}$	97.00 ± 0.43	97.78 ± 0.36
Iris	94.67 ± 3.40	95.33 ± 1.63	95.33 ± 3.40	96.00 ± 3.89
Liver	64.35 ± 2.84	$\mathbf{65.22 \pm 2.25}$	62.00 ± 2.92	$56.81 \pm 5.09^{-}$
PIMA	75.01 ± 4.64	74.47 ± 2.92	76.05 ± 3.81	74.34 ± 2.57
Vehicle	62.06 ± 1.39	$\mathbf{62.29 \pm 2.15}$	60.87 ± 1.91	$43.26 \pm 3.82^{-}$
Waveform	81.08 ± 2.87	$\mathbf{82.17 \pm 2.87}$	81.66 ± 0.96	82.00 ± 2.92
Wine	97.71 ± 2.14	97.14 ± 1.81	$\mathbf{98.29 \pm 2.29}$	97.14 ± 2.52
Yeast	56.86 ± 3.09	57.74 ± 3.20	$\mathbf{58.72 \pm 2.85}$	$46.66 \pm 5.36^{-}$
Average	80.11	80.81	79.98	75.69

Table 4. Required differences of Bonferroni multiple comparisons between the lazy methods and the well-known methods.

Data set	RD	Data set	RD
Australian	5.17	Hypothyroid	1.05
Breast-W	5.36	Iris	7.18
Cleve	9.14	Liver	6.80
CRX	4.71	PIMA	8.25
German	5.78	Vehicle	4.91
Glass	11.33	Waveform	6.01
Heart	5.87	Wine	4.47
Hepatitis	10.99	Yeast	8.40

a multi-interval value $I_m = \{I_1, I_2, \ldots, I_k\}$, where $I_1, I_2, \ldots, I_k$ are intervals. The class-conditional density of x given class c is $p(x \in I_m \mid c)$. By perfect aggregation (Theorem 1) and Lemma 2, we can estimate $\hat{p}(x \in I_m \mid c)$ by

$$\hat{p}(x \in I_m \mid c) = \frac{\alpha_m + y_{cm}}{\alpha + n_c} \tag{9}$$

where y_{cm} is the number of class c examples in the training data set whose $x \in I_m$, and $\alpha_m = \alpha_1 + \alpha_2 + \cdots + \alpha_k$. Equation (4) is a generalization of Eq. (3) and can be applied to query vectors that contain discrete or set-valued data. Note that as long as we can compute y_{cm}, we can apply this equation regardless of the data type of x in the training data set. That is, in the training data set, x can be point-valued, set-valued, or interval-valued.

7.2. Experimental results

Since there is no set-valued or interval-valued data in the UCI ML repository (Blake & Merz, 1998), we developed a technique to generate interval-valued data from point-valued data to observe the effectiveness of this new method. The idea is to form intervals by selecting pairs of point-valued continuous data with the same class label such that the difference of the values in each pair is within a certain range. By selecting an appropriate range for each data set, classifying interval data can yield much better accuracies than classifying point-valued data.

We used the ten "pure" continuous data sets from UCI ML repository to evaluate our approach empirically and compare its performance with the well-known discretization methods. Suppose we have two query vectors with the same unknown class label and x is a continuous variable in these vectors. Suppose the values of x for these vectors are X_1 and X_2, respectively, and $X_1 \leq X_2$. Then the x value of the merged vector is $[X_1, X_2]$. The idea is that if we have two different samples of the same unknown class, then it is very likely that the values of the feature of the samples in this class are within the interval bounded by the values from the two samples. For example, suppose we pick up two leaves dropped from a tree of an unknown genus and we want to use the features of the leaves, such as their length, to recognize the genus. If the length of the two leaves are 16.57 and 17.32, respectively, then we can assume that the length of the leaves of this genus is within the interval [16.57, 17.32].

However, the width of the intervals created by merging should be constrained. If two values are too close and the merged interval becomes too narrow, the number of the examples in the training data set in the interval will be too small to allow accurate estimation of $\hat{p}(x \in I \mid c)$. Therefore, we will set a minimum threshold for the interval width. If the width of the interval created by merging is smaller than the threshold, we will extend both ends of the interval to reach the minimum threshold.

On the other hand, if the difference between the two values are too large and the merged interval become too wide, the interval may contain decision boundaries and result in misclassification. Consider a simple classification task that involves a single variable x and two classes c_1 and c_2. Suppose we have two data D_1 and D_2 that have the same class label c_1 and their x values are marked in figure 13 which also shows the curves of the "true" densities of $p(x \mid c)$. If we classify D_1 and D_2 individually based on the densities, both D_1 and D_2 will be correctly classified to class c_1. If we merge D_1 and D_2, we will create an interval I as shown in figure 13. The merged data will be incorrectly classified to c_2 because $p(x \in I \mid c_1) < p(x \in I \mid c_2)$ (illustrated as the shaded areas in figure 13). Therefore, we must set a maximum threshold for the width of the merged intervals. If the width is larger than that threshold, we will partition the interval into multiple intervals with a pre-determined width S_I. In the case of figure 13, we will form two intervals I_1 and I_2 that contain D_1 and D_2 respectively. The merged query vector will have a multi-interval value $\{I_1, I_2\}$.

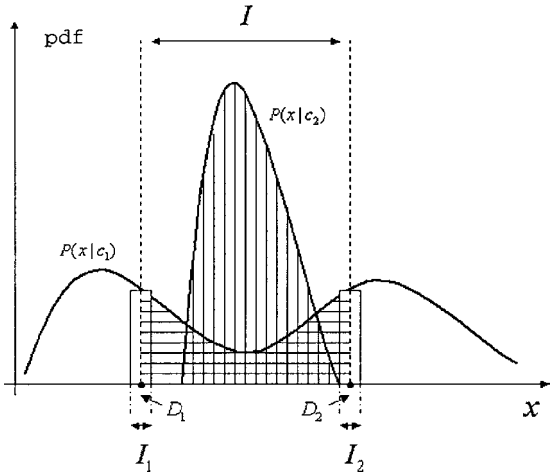

Figure 13. Merging two widely separated samples may result in misclassification.

In the experiment, the thresholds are determined empirically as follows. The threshold of the minimum width for each variable is equal to the width of "fifty-bin" for large data sets (size > 350), "twenty-bin" for medium-sized data sets (size between 350 and 150), and "fifteen-bin" for small data sets. The threshold of the maximum width is equal to "four-bin", because we investigated the experiments in Section 5.2 and concluded that when the number of bins is greater than four bins, the accuracies will reach a plateau for most data sets. The value of S_I is equal to the threshold of the minimum width of the variable.

For each data set selected for the experiment, we randomly chose twenty percent of data from each class for testing and the remainder for training. Then we randomly picked pairs of data from the test set and merged them into interval data by applying the method described above. The merged data are classified by the "lazy interval" approach described in Section 7.1 and the resulting accuracy is reported. This procedure is repeated ten times to obtain the average and the standard deviation of the accuracies for each data set. Table 5 gives the results. We also extract the best results for each data set from Tables 2 and 3 for comparison. For all data sets, "lazy interval" outperforms the best result of discretization significantly. This result reveals that the "lazy interval" approach is useful for classifying interval data.

The improvement of the classification accuracies by the lazy interval approach can be attributed to the pre-processing step that merges the point-valued data into the multi-interval data. The merging reduces the probability of "*Bayes error*." Bayes error occurs when a query vector is classified to a wrong class when the Bayes decision rule is applied. The region of Bayes error for a class C and a variable x is the region in the domain of x where Bayes error occurs.

Given a pair of query vectors to be merged, their values can be either inside or outside of the region of Bayes error. Therefore, we have three possible combinations. The first combination is that both query vectors are outside of the region of Bayes error. Then the merged interval data will be correctly classified. In the second combination, we have both query vectors in the region of Bayes error, and the classification result will be incorrect. Both

Table 5. Comparison of the accuracies achieved by the naive Bayesian classifier that classifies merged interval data and the best accuracies achieved by the naive Bayesian classifier that classifies point data.

Data set	Lazy interval	The best of Table 2 and 3
Breast-W	98.60 ± 1.31	96.00 ± 3.08
German	78.40 ± 2.76	75.30 ± 1.42
Glass	67.39 ± 6.53	67.12 ± 3.60
Iris	100.00 ± 0.00	97.00 ± 4.33
Liver	68.29 ± 5.49	65.22 ± 2.25
PIMA	81.71 ± 4.22	76.18 ± 4.84
Vehicle	64.22 ± 5.36	62.29 ± 2.51
Waveform	89.45 ± 2.52	82.17 ± 2.87
Wine	99.44 ± 1.67	97.78 ± 1.67
Yeast	64.66 ± 3.04	58.72 ± 2.85
Average	81.22	77.78

of the above combinations yield the same result as classifying the data without merging. In the third combination, one of the query vector is in the region while the other is not. In this case, without merging, one query vector will be classified correctly while the other will be misclassified. But with merging, it is more likely that the merged query vector will be correctly classified, and as a result, the classification accuracy will be boosted.

We can show that this is the case. Consider again a simple classification task that involves a single variable x and two classes c_1 and c_2 with their p.d.f. curves given in figure 14. D_1 and D_2 are two query vectors of class c_1. D_2 is in the region of Bayes error but D_1 is not.

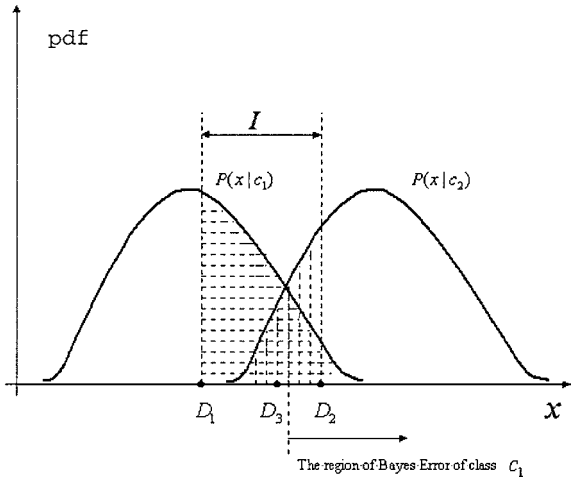

Figure 14. The area under the p.d.f. curve in the region of Bayes error is usually small and therefore merging can improve the classification accuracy.

If we classify each of them based on their conditional densities, D_2 will be misclassified. However, if we merge D_2 with D_1, the classification will be based on $p(x \in [D_1, D_2] \mid c_1)$ and $p(x \in [D_1, D_2] \mid c_2)$, the areas between the interval $[D_1, D_2]$ under the curves of $p(x \mid c_1)$ and $p(x \mid c_2)$ respectively. For x value in the region of Bayes error, such as D_2, since D_2 is supposed to be of class c_1, it is likely that both $p(x \mid c_1)$ and $p(x \mid c_2)$ are low and their difference is small. On the other hand, if x is outside of the region, such as D_1, then it is quite likely that $p(x \mid c_1) \gg p(x \mid c_2)$. Therefore, chances are that we have $p(x \in [D_1, D_2] \mid c_1) > p(x \in [D_1, D_2] \mid c_2)$, as illustrated in figure 14, and hence the merged query vector will be classified correctly. A counterexample is the pair D_2 and D_3. Merging this pair will yield incorrect result. But it is unlikely that we select such a pair of query vectors because $p(x = D_3 \mid c_1)$ must be relatively low.

8. Conclusions

This paper reviews the properties of Dirichlet distributions and describes its implications and applications to learning naive Bayesian classifiers. The results can be summarized as follows:

- Perfect aggregation of Dirichlets ensures that a naive Bayesian classifier with discretization can effectively approximate the distribution of a continuous variable.
- A discretization method should partition the domain of a continuous variable into intervals such that their cut points are close to decision boundaries and their width is sufficiently large. It turns out that this requirement is not difficult to achieve the above requirement for many data sets and as a result, a wide variety of discretization methods can have similar performance regardless of their complexities.
- We presented a new lazy discretization method, which is derived directly from the properties of Dirichlet distributions, and showed that it works equally well compared to well-known discretization methods. These results justify the selection of Dirichlet priors and verify our analysis.
- We also extend the method to allow a naive Bayesian classifier to classify set and interval data.

We plan to investigate whether our analysis can provide any new insight on handling continuous variables in general Bayesian networks as our future work.

Acknowledgments

The research reported here was supported in part by the National Science Council in Taiwan under Grant No. NSC 89-2213-E-001-031.

Notes

1. Though the document of `heart` states that all of its attributes are continuous, attribute 2, 3, 6, 7, 9, 11, 12, 13 are actually binary or discrete.

2. The accuracies were obtained from five-fold cross-validation tests where continuous variables are handled by the lazy ten-bin discretization method described in Section 6.

References

Almond, R. (1995). *Graphical Belief Modelling*. New York: Chapman and Hall.

Azaiez, M. N. (1993). Perfect aggregation in reliability models with Babyesian updating. Ph.D. thesis. Department of Industrial Engineering, University of Wisconsin-Madison, Madison, Wisconsin.

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Cestnik, B., & Bratko, I. (1991). On estimating probabilities in tree pruning. In *Machine Learning—EWSL-91, European Working Session on Learning* (pp. 138–150). Berlin, Germany: Springer-Verlag.

Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning, 29*, 103–130.

Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Machine Learning: Proceedings of the 12th International Conference (ML '95)*. San Francisco, CA: Morgan Kaufmann.

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

Fayyad, U. M., & Irani, K. B. (1993). Multi-interval discretization of continuous valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI '93)* (pp. 1022–1027).

Heckerman, D. (1998). A tutorial on learning with Bayesian networks. In M. I. Jordan (ed.), *Learning in Graphical Models* (pp. 301–354). Boston: Kluwer Academic Publishers.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning, 11*, 63–90.

Iwasa, Y., Levin, S., & Andreasen, V. (1987). Aggregation in model ecosystem: Perfect aggregation. *Ecological Modeling, 37*, 287–302.

John, G., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI '95)* (pp. 338–345).

Kohavi, R., & Sahami, M. (1996). Error-based and entropy-based discretization of continuous features. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD '96)* (pp. 114–119). Portland, OR.

Langley and Thompson. (1992). An analysis of Bayesian classifier. In *Proceedings of the 10th National Conference on Artificial Intelligence* (pp. 223–228). Portland, OR: AAAI Press.

McClave, J. T., & Dietrich, F. H. (1991). *Statistics*. San Francisco: Dellen Publishing Company.

Monti, S., & Cooper, G. F. (1999). A bayesian network classifier that combines a finite mixture model and a naive bayes model. In *The fifteenth Conference on Uncertainty in Artificial Intelligence (UAI '99)* (pp. 447–456).

Spector, P. (1990). *An Introduction to S and S-PLUS*. Duxbury Press.

Wilks, S. S. (1962). *Mathematical Statistics*. New York: Wiley and Sons.

Wong, T.-T. (1998). Perfect aggregation in dependent Bernoulli systems with Bayesian updating. Ph.D. thesis, Department of Industrial Engineering, University of Wisconsin-Madison, Madison, Wisconsin.

Received November 9, 2000
Revised May 17, 2002
Accepted May 17, 2002
Final manuscript May 17, 2002