Bayesian Classification for Set and Interval Data

Hung-Ju Huang
Department of Computer and
Information Science
National Chiao-Tung University
Hsinchu City 300, Taiwan
gis85563@cis.nctu.edu.tw

Chun-Nan Hsu
Institute of
Information Science
Academia Sinica,
Taipei City, Taiwan
chunnan@iis.sinica.edu.tw

Abstract— **Learning naive Bayesian classifiers is an important approach to probabilistic induction. However, no study has been done on naive Bayesian classifiers when a query vector includes interval-valued data, and little is known about how a set of query vectors from the same unknown class can be accurately classified. In this paper, we present a new training approach to the problems above. This approach is based on the *"perfect aggregation"* property of the Dirichlet distribution, which is usually assumed to be the prior of the variables in a Bayesian classifier. The experimental results show that when we merge an appropriate number of query vectors with the same unknown class and the interval-valued data are formed, the acccuracies of a trained naive Bayesian classifier can be promoted significantly. This paper also reports a successful application of our approach in speaker recognition.**

Key Words: Naive Bayesian Classifier, Interval Query, Machine Learning, Data Mining

I. Introduction

Learning naive Bayesian classifiers is an important approach to probabilistic induction. In spite of its simplicity, naive Bayes constantly outperforms competing algorithms in the experiments reported in the literature. Remarkably, in KDD-CUP-97, two of the top three contestants are based on naive Bayes [1]. Meanwhile, Domingos and Pazzani [2] reported an experiment that compared naive Bayes with several classical learning algorithms such as C4.5 with a large ensemble of real data sets. The result also showed that naive Bayes can significantly outperform those algorithms. Hence, the naive Bayesian classifier is becoming a popular tool for classification. Naive Bayes can handle discrete variables and continuous variables assuming that their priors are Dirichelet distribution and Normal distribution, respectively [3]. It has been shown that when a continuous variable is not normal, the performance will be inferior to discretization.

Several researchers have studied how to handle continuous variables for Bayesian classifiers [3–7]. However, no study has been done on naive Bayesian classifiers when a query vector including interval-valued data [8], and little is known about how a set of query vectors from the same unknown class can be accurately classified. The relation between these two problems can be illustrated by the following example. Suppose we pick up two leaves dropped from a tree, we would like to use their features, such as their length, to classify what kind of tree they were on. In general, the length of the two leaves will not be equal. Conventionally, we can classify each individual leaf by a classifier. But if the classification results are not the same, we will have to come up with a method to resolve the difference. Alternatively, we can merge each feature of the two leaves to form a new interval-valued data, and then classify the merged data.

Consider a data set that comes from three different classes. We want to classify two query vectors. Assuming that we do not have any prior knowledge about the class distribution. Hence, suppose we randomly assign a class label to each vector. The expected accuracy will be $(1*1/3*1/3+2*0.5*1/3*2/3+2*0*2/3*2/3=1/3)$. On the other hand, suppose we know that the two vectors belong to the same unknown class, and randomly assign their class. The expected accuracy will still be $(1*1*1/3+1*0*2/3=1/3)$, regardless of our knowledge that they come from the same class. This shows that a classifier must deliberately take advantage of that knowledge or the knowledge will not improve the expected accuracy.

In this paper, we present a method that allows a naive Bayesian classifier to have the abilities of processing interval-valued data. This method is then extended to classify merged query vectors when we know these vectors have the same unknown class label. The key of our approach is based on our study on the Dirichlet distribution and its properties. A discrete variable as well as a discretized continuous variable in a naive Bayesian classifier are usually assumed to have a Dirichlet prior. Perfect aggregation of Dirichlets implies that we can estimate the class-conditional probabilities of discretized intervals regardless of how other region of the domain of the continuous variable is discretized. Those are the reasons of why we can process multiple interval-valued data. The experimental results show that when we use interval data which were formed by merging an appropriate number of query vectors with the same unknown class label, the accuracy of a naive Bayesian classifier will be promoted significantly.

II. Preliminary

A. Dirichlet Distribution and Perfect Aggregation

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

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

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 distributions and usually denoted $Beta(\alpha_1, \alpha_2)$. critical to our approach. These properties greatly simplify the computation of the moments of the Dirichlet distribution in Bayesian analysis. Suppose random vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_k)$ has a Dirichlet distribution $D_k(\alpha_1, \alpha_2, \ldots, \alpha_k; \alpha_{k+1})$, then by [9] any subvector $(\theta_{n1}, \theta_{n2}, \ldots, \theta_{nm})$ of θ has an m-variate Dirichlet distribution $D_m(\alpha_{n1}, \alpha_{n2}, \ldots, \alpha_{nm}; \alpha - \sum_{j=1}^{m} \alpha_{nj})$. We call this *subvector lemma*. Also by [9], 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 $\alpha = \alpha_1 + \alpha_2 + \cdots + \alpha_{k+1}$. This is called *sum-of-subset lemma*.

Another important property of Dirichlets is that a Dirichlet distribution is *conjugate* to the multinomial sampling [10]. 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 turning to be outcome j. When the prior distribution $p(\boldsymbol{\theta})$ is a Dirichlet distribution with parameters α_j for $j = 1, 2, \ldots, k+1$, and the likelihood function $L(D|\boldsymbol{\theta})$ follows a multinomial distribution, then the posterior distribution $p(\boldsymbol{\theta}|D)$ 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.

The expected value for θ_j given D is $E[\theta_j|D] = \frac{\alpha_j + y_j}{\alpha + n}$, for $j = 1, 2, \ldots, k+1$. This expression can be rewritten as follows:

$$\begin{aligned} E[\theta_j|D] &= \frac{\alpha_j + y_j}{\alpha + n} = \frac{\alpha}{\alpha + n}\frac{\alpha_j}{\alpha} + \frac{n}{\alpha + n}\frac{y_j}{n} \\ &= wE[\theta_j] + (1 - w)\frac{y_j}{n}, \end{aligned}$$

where $w = \frac{\alpha}{\alpha + n}$. Note that $E[\theta_j]$ and y_j/n are the prior and the sample means of θ_j, respectively. Hence, w and $1 - w$ can be thought of as the weights of prior and sample means, respectively, and the weights for all j are all identical. This reveals the advantage that a Bayesian analysis considers both prior information and training data.

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 $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_k)$ is a Dirichlet distribution $D_k(\alpha_1, \alpha_2, \ldots, \alpha_k; \alpha_{k+1})$. A straightforward application of the Bayesian approach is to use the training data D to update the prior distribution to obtain the posterior distribution $f(\boldsymbol{\theta}|D)$, 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 $f(q|D)$ is derived from $f(\boldsymbol{\theta}|D)$. By subvector lemma and sum-of-subset lemma, $f(q|D)$ is a Beta distribution:

$$Beta(\textstyle\sum_{j \in \Phi}(\alpha_j + y_j), \alpha + n - \sum_{j \in \Phi}(\alpha_j + y_j)) \quad (1)$$

However, by the properties of the Dirichlet distribution, there exists an alternative and simpler way to compute $f(q|D)$. We can first derive the prior distribution for the probability of interest $f(q)$ from the prior distribution of $f(\boldsymbol{\theta})$. 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 $f(q|D')$.

We can show that in general it is always the case that $f(q|D) = f(q|D')$. Since the multinomial likelihood function $L(D|\boldsymbol{\theta})$ implies that the trials for obtaining data set D are all independent, the likelihood function $L(D'|q)$ will follow a binomial distribution. By sum-of-subset lemma, the prior distribution of $f(q)$ has a beta distribution $Beta(\sum_{j \in \Phi} \alpha_j, \alpha - \sum_{j \in \Phi} \alpha_j)$ when $\boldsymbol{\theta}$ has a Dirichlet distribution. Since the beta distribution is conjugate to the binomial sampling, the posterior distribution $f(q|D')$ will have a Beta distribution with parameters $\sum_{j \in \Phi}(\alpha_j + y_j)$ and $\alpha + n - \sum_{j \in \Phi}(\alpha_j + y_j)$. This is exactly the same as Equation (1). This property is always true for the Dirichlet distribution and is first derived by [11], called *"perfect aggregation"* [12,13].

For example, suppose that we are interested in the probability of showing 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 $f(q|D)$ 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.

B. Naive Bayesian Classifier

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

$$p(c|\mathbf{x}) \propto p(c) \prod_{x \in \mathbf{x}} p(x|c), \quad (2)$$

where x is a variable in $\mathbf{x}$. $p(x|c)$ is the *class-conditional density* of x given class c. Let $\boldsymbol{\theta}$ denote the vector whose elements are the parameters of the density of $p(x|c)$. In a Bayesian learning framework, we assume that $\boldsymbol{\theta}$ is an uncertain variable [10] and can be learned from a training data set. This estimation is at the heart of training in a naive Bayes.

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, $p(x = X_j|c) = \theta_j$ and $\sum_j \theta_j = 1$. Now, let $\boldsymbol{\theta} \equiv (\theta_1, \theta_2, \ldots \theta_k)$. We choose a Dirichlet distribution with parameters $\alpha_1, \ldots, \alpha_{k+1}$ as the prior for $\boldsymbol{\theta}$. Given a train data set, we can update $p(x = X_j|c)$ by its expected value:

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

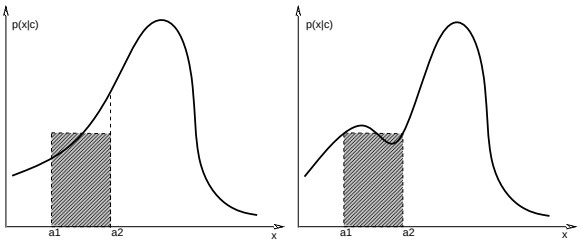

Fig. 1. Partition independence assumption

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 $\boldsymbol{\theta}$ 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 Bayes.

In practice, we usually choose the Jaynes prior [14] $\alpha_j = \alpha = 0$ for all j and have $\hat{p}(x|c) = \frac{y_{cj}}{n_c}$. However when the training data set is too small, this often yields $\hat{p}(x|c) = 0$ 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* [15].

If x is a continuous variable, 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 classifying in a naive Bayes with discretization is to use $\hat{p}(x \in I_j|c)$ as an estimate of $\hat{p}(x|c)$ in Equation (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 holds for all well-known discretization methods, including ten-bin, entropy-based, etc. See [4] for a comprehensive survey.

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

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. $f(x|c)$, 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. are identical.

C. Implications of Perfect Aggregation

An implication of perfect aggregation is that to estimate the posterior probability of a union of disjoint events, there is no need to estimate the probabilities of individual events. 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 only concern 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 property can simplify the training effort in identifying the outcome of an observation in D. These implication allow us to derive a lazy discretization method and a multi-interval classifier for naive Bayes.

In our previous work [16], we proposed a lazy discretization method for continuous variables. This 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 a test datum for each variable. That is, it creates one interval (denoted as I) and leaves the other region untouched. From the training data, we can estimate $\hat{p}(x \in I|c)$ by the expression given in Section II-B and use this estimate to classify the test datum. This method can invoke different instantiations to determine the cut points. For example, we can select a pair of cut points such that the value of x is in the middle and the distance between the cut points is the same as the width of the intervals created by ten-bin[1] We will call it "lazy ten-bin." Similarly, we can have "lazy entropy," "lazy bin-log," etc.

This discretization method is derived from the perfect aggregation and other properties of the Dirichlet distribution. Suppose that partition independence assumption holds. Then by the sum-of-subset lemma of the Dirichlet distribution, "$x|c$" will have a Beta prior with parameters $\alpha \int_I p(x|c)dx$ and $\alpha(1 - \int_I p(x|c)dx)$, where c is a class and α is an equivalent sample size. By perfect aggregation, we can estimate $\hat{p}(x \in I|c)$ 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. This way, it may simplify the training effort.

In order to show that the lazy ten-bin can perform as well as well-know discretization methods, we empirically compared lazy ten-bin, ten-bin, entropy-based, and a Gaussian version of naive Bayes on ten real data sets from UCI machine learning repository [17]. Table I gives the average results of the ten real datasets with different discretization methods. The detail can be found in [16].

III. Classifying Set and Interval Data

A. Set and Interval Data

We begin with some necessary definitions.

Definition 1: A variable is said to have **Set Values** if its value is a set.

[1]Ten-bin is a discretization method that divides the domain of a continuous variable into ten equal width bins.

TABLE I

AVERAGE ACCURACIES OF NAIVE BAYES WITH DIFFERENT DISCRETIZATION METHODS.

DATASET	LAZY TEN-BIN	TEN BINS	ENTROPY	GAUSSIAN
AVERAGE	75.99	75.27	75.76	69.93
WIN:LOSS	–	8:2	5:5	9:1

For example, a variable X with its value equal to $\{a, b, c\}$ is said to have a set value, where a, b, c are some possible states of X.

Definition 2: A variable is said to have interval values if its value can be an interval; a vector is a piece of interval data if one of its element variable has an interval value.
For example, V_1 includes a variable A, and $A = [20.5, 38.5]$. Then V_1 is an interval data.

Definition 3: When an element of a vector has a set value which consists of interval members, this vector is a **Multi-Interval Data**.
For example, V_2 includes a variable B, and $B = \{[20.5, 38.5], [50.5, 60.5]\}$, then V_2 is a multi-interval data.

B. Training and Classification for Set and Multi-Interval Data

In the Section II-A, we described the approach of Lazy Discretization. If a query vector contains interval data, we can simply let the discretized interval I be the given interval and no more discretization is necessary. That is a direct extension of Lazy Discretization for Interval Data. Furthermore, when we are interested in several different segments of a variable simultaneously, the class-conditional density of our interested segment given class c can be done by $p(x \in I_m|c)$, where $I_m = \{I_1, I_2, ..., I_k\}$, and k is the number of the segments we are interested in. By the perfect aggregation and the sum-of-subset lemma of Dirichlets, we can estimate $\hat{p}(x \in I_m|c)$ by the Equation (4).

$$\hat{p}(x \in I_m|c) = \frac{\alpha_m + y_{cm}}{\alpha + n_c}. \quad (4)$$

It is the extension of Equation (3), where y_{cm} is the number of class c examples whose $x \in I_m$, and the $\alpha_m = \alpha_1 + \alpha_2 + ... + \alpha_k$.

So, we can handle a multi-interval data by Equation 4. When the variable is a discrete variable and has set value, the Equation (4) is also can be used. Note that we still assume the query examples to contain discrete and continuous values are usually.

To speed up the estimation of $\hat{p}(x \in I|c)$ in our implementation, we can divide each domain of continuous variable into a large number of equal-width bins. Then we count the number of training examples falling in each bin for a given class c and save them in a table. After that, we can calculate an approximate value of $\hat{p}(x \in I|c)$ by examining the table. The larger number of bins that we divide in advance, the closer the estimated will be to the real value. In our experiments, we divided each continuous variables into one thousand equal-width bins.

C. Merging Point Data into Set and Interval Data

Now, we will describe that how more than one query vectors can be merged and classified a single query vector when we know these vectors have the same unknown class label, then we can form interval-value data. Consider the tree classification problem in Section I. If the difference of the two leaves' length is not very large, it may be reasonable to assume that this kind of trees have leaves with length within the interval formed by the length of the two leaves that we have got. Then, we can use our multi-interval query method that was discussed in Section III-B to classify the merged query vector. So, when we observe the set of query vectors with the same unknown class label, we will form an interval for each continuous feature that is bounded by the minimum and the maximum values of each feature in that set.

But we can not use the interval for query without further consideration, because there are some situations that may not improve the performance. Hsu and Huang [16] concluded that to avoid performance degradation, a discretization method should partition the domain of a continuous variables into intervals such that their cut points are close to decision boundaries to minimize the distortion due to the discretization and their width should be sufficiently large to cover sufficiently many training examples. Similar reasoning applies to this case. If the interval is too narrow, the number of examples in the training data set in that interval will be too small to allow accurate estimation of $\hat{p}(x \in I|c)$. To avoid this, we will set a minimal interval threshold. If the interval is smaller than that threshold value, we will extend both ends of the interval to reach the minimal threshold.

If the interval is too wide, it may contain decision boundaries, and degrade the performance. Consider the two conditional distributions as shown in Figure 2, and assume data D_1 and D_2 have the same class label C_1. Based on our previous discussion, we will form an interval I, but the interval I will include decision boundaries. If we classify the data D_1 and D_2 individually, the results will be correct (both D_1 and D_2 will be classified to class C_1). But if we use the interval I to classify them, the result will be wrong (Both D_1 and D_2 will be classified to class C_2) because the area under the p.d.f given C_1 is smaller than the area of the p.d.f given C_2.

To avoid this, we must set a maximal interval threshold. If an interval is larger than that threshold, we will divide it into multiple intervals, based on a suitable width S_I. In the case of Figure 2, we will form two intervals I_1 and I_2 such that one of them includes data D_1 or D_2 and the width of them were set as the width of S_I. When we use the

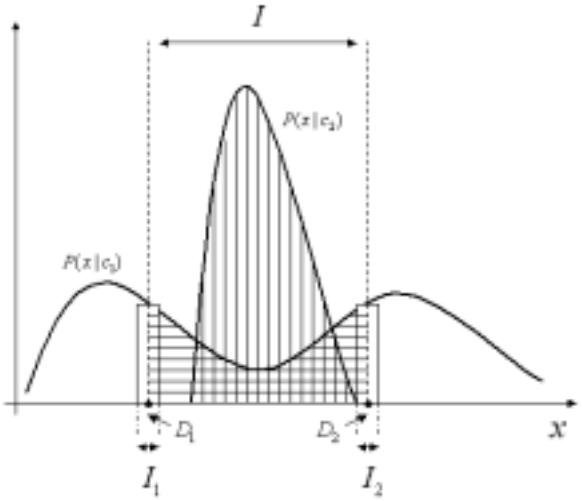

Fig. 2. Two conditional distribution

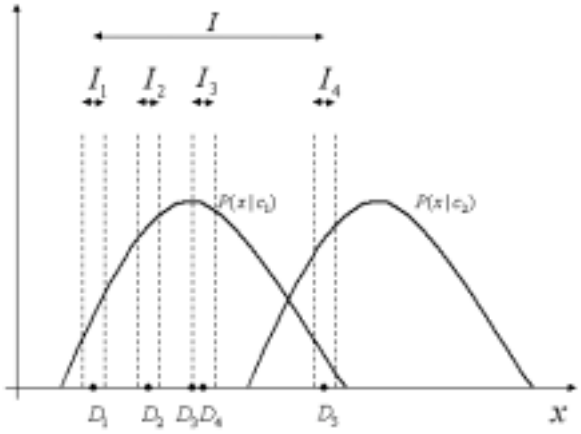

Fig. 3. Two conditional distribution

intervals I_1 and I_2 for the query, the result will be correct in this case.

However, consider the conditional distributions as shown in Figure 3 and assume that interval I is too wide. If there are other query examples whose values fall in the region of interval I, and we only use the intervals (I_1 and I_4) created by the boundaries(D_1 and D_5), then we will lose the information from other query examples (D_2, D_3, D_4). To avoid this, we also create other intervals (the interval's width are set to be S_I too) in order to include the information of the other data. For example, in order to include all five data in Figure 3, we discrete the interval I into four intervals (I_1, I_2, I_3, and I_4), then we can use the information from D_2 to D_4.

In all our experiments, we set the minimal interval threshold of each feature equal the width of "fifty-bin", which was derived by dividing the domain of each continuous variables in the training set into fifty equal-width bins for all the data sets, except the data set "Iris". We set the minimal interval threshold for the data set "Iris" equal to the width of "fifteen-bin", because the size of this data set is too small. We set the maximize interval threshold equal to "four-bin",[2] We set the width of S_I equal to "fifty-bin" by taking the maximum number of bins in the experiment of [16].

In summary, to improve performance, when an interval is too narrow, we will extend it to as wide as S_I. If it is too wide, we will divide it into multiple intervals and the width of each subinterval is also set to S_I. Then we can use our multi-interval classifier.

IV. Experimental Results and Discussion

A. Experiment

To observe the effectiveness of our approach to real world problems, we select five data sets from the UCI machine learning repository [17] for our experiments. We first partitioned each data set into several subsets (the number of subsets is equal to the number of classes in that data set) according to the data's class label. In each subset, we randomly selected twenty percentage of each class for test and the remnant for training. In the test set, we merged two test vectors from the same class according to the approach described in Section III-C. We also used the same sets (training and test) to obtain the results of the "lazy ten-bin" which was proposed by [16]. In this case, only one query vector is classified at a time.

We repeated the experiment ten times and reported the average and the standard deviation of the accuracies. For comparison, we also list the experimental results of "Ten-bin", "Entropy" and "Gaussian" for the same set of data sets. However, the accuracies reported here were obtained by running five-fold cross validations. Table II gives the result, which reveals that our method is significantly better than the "lazy ten-bin" in all the data sets.

B. Discussion

We will show that a classifier must deliberately take advantage of that knowledge or the knowledge will not improve the expected accuracy, and this is the case in general. Consider a "random" classifier which randomly guess the class of query vectors. Suppose that two query vectors were classified individually with this "random" classifier into one of n classes. The expected value of the accuracy can be derived as follows.

$$\begin{aligned} E_1 &= \binom{2}{2}\frac{2}{2}(\frac{1}{n})^2 + \binom{2}{1}\frac{1}{2}\frac{1}{n}\frac{n-1}{n} + \binom{2}{0}\frac{0}{2}(\frac{n-1}{n})^2 \\ &= \frac{1}{n^2} + \frac{1}{n}\frac{n-1}{n} = \frac{1}{n^2} + \frac{n-1}{n^2} \\ &= \frac{n}{n^2} = \frac{1}{n} \end{aligned}$$

Now, suppose we know that the two vectors actually belong to the same class and classify them together using the "random" classifier. The expected value of the accuracy is:

$$E_2 = \binom{1}{1}\frac{2}{2}\frac{1}{n} + \binom{1}{0}\frac{0}{2}\frac{n-1}{n} = \frac{1}{n}$$

[2]We investigated the experiments in [16] and found that when the number of discretized equal-width bins is larger than four bins, the accuracies reach plateau situations for most data sets.

TABLE II
ACCURACIES OF NAIVE BAYES WITH DIFFERENT APPROACH

DATASET	MERGE	LAZY TEN-BIN	TEN BINS	ENTROPY	GAUSSIAN
BREAST	98.60±1.31	94.39±2.46	94.37±2.16	94.37±1.93	92.79±2.61
IRIS	100.00±0.00	97.00±4.33	94.67±3.40	95.33±1.63	96.00±3.89
PIMA	81.71±4.22	76.12±3.51	75.01±4.64	74.47±2.92	74.34±2.57
SONAR	85.71±6.68	78.57±6.12	75.55±3.8	72.69±5.88	69.21±5.88
VEHICLE	64.22±5.36	61.08±2.71	62.06±1.39	62.29±2.15	43.26±3.82
AVERAGE	86.05	81.43	80.33	79.83	75.37
WIN:LOSS	-	5:0	5:0	5:0	5:0

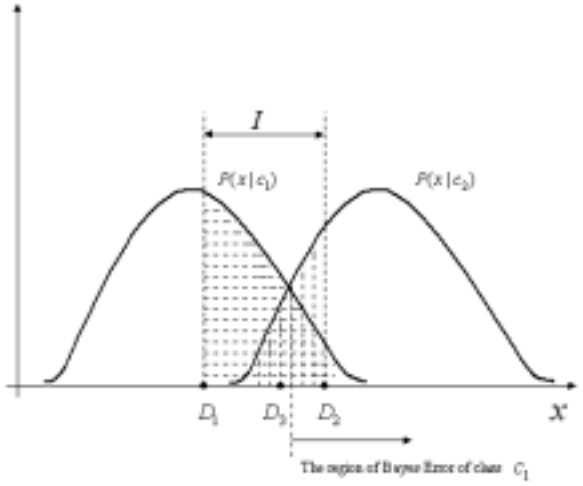

Fig. 4. Two conditional distribution

The expected value of the two cases are equal. This result implies that the information that we know the two vectors come from the same class seems not helpful for improving the accuracy. However, why were the accuracies improved significantly in all the data set in our experiments?

Consider the two class-conditional density functions of a variable x given class C_1 and C_2 as shown in Figure 4. Assume the data D_2 belongs to class C_1 actually and falls in the region of Bayes error[3] of class C_1.

If we classify data D_2 individually based on the Bayes decision rule, D_2 will be classified as C_2 which is incorrect. But if we classify data D_2 with another data D_1 which belongs to class C_1 actually and is far away from the region of Bayes error of C_1. Now, if we classify data D_2 and D_1 together based on the area under C_1 and C_2 within the interval I, we can obtain the correct result. It is because the area under the p.d.f given C_1 is larger than the area of the p.d.f given C_2. Hence, if a data falls in the region of Bayes error of its actual class and the data can be classified with another data having the same class, it is more likely that the data will be classified correctly. We called this situation as $Case1$. But if data D_2 is classified with unsuitable data then the area under the p.d.f given C_1 is smaller than the area of the p.d.f given C_2, the classified result of data D_2 still is wrong. We called this situation as $Case2$. For example, data D_2 is classified with data D_3 which also belong to class C_1 actually and is near the region of Bayes error of C_1, then the $Case2$ is occurred.

In general, if a class can be differentialed, the region of Bayes error of this class usually appears on the region which has lower probability. Hence, in our experiments if one data falls in the region of Bayes error of its actual class, the probability of the another data with the same class label which is near or falls in the same region is also lower. So, the frequency of $Case1$ occurred that is often larger than the frequency of $Case2$. That is why the accuracies were improved significantly in all the data set in our experiments.

In the experiment of Section IV-A, we only merged two test-data have the same unknown class label for classification. We also study the effect of our method, if we merge more than two data that have the same unknown class label in our approach. We selected a larger data set "waveform" from UCI, and repeated the experiment in Section IV-A with different number of test-data being merged (from 2 to 50). Figure 5 shows the results. The first value in Figure 5 was generated by considering only one test-data and the lazy ten-bin was applied. The result show that as the number of the merged test-data increased the curve rise and then reach a peak before it drops gradually. We will try to explain this phenomenon.

In the first phase, the curve rise, because with the number of merged test-data increased the occurrence of $Case1$ will be more often than $Case2$. That is because with the number increased the probability that both end points of the interval I fall in the same region of Bayes error of a class will descend. However, why dose the accuracy drops gradually? Recall that if the query interval I is too wide, it may degrade a performance which was mentioned in Section III-C. Hence, we will discretize it into multiple intervals. This is to avoid the situation as shown in Figure 2. But when the interval I in Figure 2 include too many other test-data, the combination of those multiple intervals which we will discretize may approach to the original interval I. Hence, the disretization will not helpful to when there are too many test-data. So, if we merge too many test-data, the situation like the above may occur. That explains why the curve drops gradually.

When the curve reaches a peak, then it is the optimum region where the number of test-data we should merge.

[3]Bayes error is the probability that a sample is assigned to the wrong class when the Bayes decision rule is applied. The region of Bayes error of a class C for a variable x is the region in the domain of x where x will be misclassified if x is of class C.

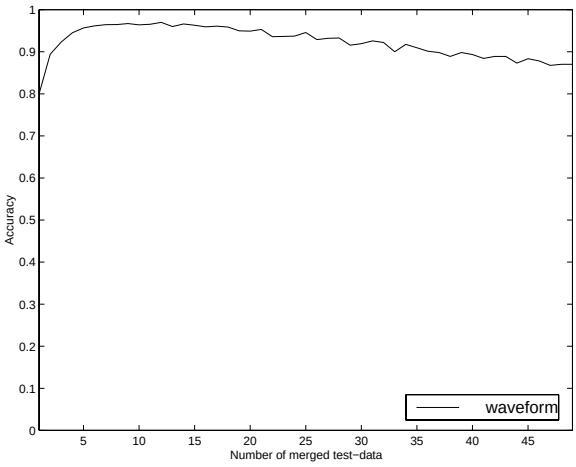

Fig. 5. Accuricies of different number of merged test-data

Different data set has different distribution and the optimum number of mergence is also different. The optimum number is case dependent.

V. Application to speaker Recognition

The proposed method was applied to a text-independence and close-set speaker recognition task. This task is particularly relevant to our approach because in this task, we usually know which set of feature vectors is from the same unknown speaker. Since a large number of feature vectors can be extracted from a short speech sentence, and obviously those vectors must come from the same speaker, a speaker recognition system should take advantage of this information.

The database for the experiments reported in this paper is a subset of the MAT2000Edu which is a speech database of Mandarin Chinese collected from many colleges in Taiwan. We used the speech data that were recorded in the National Chiao-Tung University. All speech signals were digitally recorded in a laboratory using a personal computer with a 16-bit sound blaster card and a head-set microphone. The sampling rate was 16 kHz. A 30-ms Hamming windows was applied to the speech every 10 ms. For each speech frame a 12th-order linear predictive and a log energy analysis were performed. A feature vector for query was consisted of the twelve linear predictive parameters and a log energy parameter of a frame. There are more than ten sentences that were recorded from each speaker and each sentence could be extracted more than four thousand feature vectors. In our experiments, all the original feature vectors were considered to use and no silence-removing algorithm was applied. We randomly selected 2 to 6 speakers from the subset.

The evaluating procedures were described in the follow. We set the number of merged test-data to forty, which was determined empirically, and the other parameters (the minimal interval threshold, the maximal interval threshold, and the S_I) were set the same as the experiments in Section IV-A. We randomly selected five sentences for each speaker, one for test and the others for training. And we randomly selected one thousand feature vectors from each sentence. Hence, for each speaker there were four thousand feature vectors for training and one thousand feature vectors for test. We ran the procedures ten times of each experiment on different number of speakers and reported the average and the standard deviation of the accuracies. We also showed the results of the method "lazy ten-bin". Table III gives the results. The results show that our method outperformed the "lazy ten-bin" significantly in the experiments, especially when the number of speakers is increased.

VI. Conclusions and Future Work

A discrete variable as well as a discretized continuous variable in a naive Bayesian classifier are usually assumed to have a Dirichlet prior. Perfect aggregation of Dirichlets implies that we can estimate the class-conditional probabilities of discretized intervals regardless of how other region of the domain of the continuous variable is discretized. Because of perfect aggregation of Dirichlets, we have presented a new approach that could process multiple interval queries of naive Bayes classifiers. In order to form interval data, we merged more than one query vectors from the same unknown class to one. Experimental results against standard data sets from UCI repository show that when we merged two query vectors with the same unknown class label, our approach can outperform traditional approach which only one query vector at a time. The approach can be applied successfully to the task of speaker recognition. We show that by merging an appropriate number of query vectors with the same unknown class, the acccuracies of naive Bayesian classifiers will be promoted significantly. Hence, if query vectors include interval data or the knowledge of which data come from the same unknown class, our approach will be suitably applied.

Although our approach improves the accuracy of naive Bayes classifier when we merged more than one query vectors, some parameters need to be set to obtain optimal results. But the setting of parameters is case dependant. In our experiments, those parameters were determined empirically. Hence, our future work includes to develop a approach to set those parameters automatically, and we also plan to investigate whether our approach can be applied on general Bayesian classifiers.

Acknowledgements

The research reported here was supported in part by the National Science Council of ROC under Grant No. NSC 89-2213-E-001-031. The speech data set was the courtesy of the Speech Processing Laboratory, Department of Communication Engineering, National Chiao Tung University, Taiwan.

References

[1] Ismail Parsa. KDD-CUP 1997 presentation, 1997.
[2] Pedro Domingos and Michael Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.
[3] George John and Pat Langley. Estimating continuous distributions in Bayesian classifiers. In *In Proceedings of the Eleventh*

TABLE III

ACCURACIES OF THE APPLICATION IN SPEAKER RECOGNITION

SPEAKERS	MERGE	LAZY TEN-BIN
2	99.60±1.13	80.07±1.15
3	99.20±1.07	69.74±2.12
4	95.50±2.01	57.87±0.93
5	95.36±2.17	52.60±0.87
6	92.83±2.72	49.63±0.67
AVERAGE	96.50	61.98
WIN:LOSS	-	5:0

Annual Conference on Uncertainty in Artificial Intelligence (UAI '95), pages 338–345, 1995.

[4] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Machine Learning: Proceedings of the 12th International Conference (ML '95)*, San Francisco, CA, 1995. Morgan Kaufmann.

[5] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI '93)*, pages 1022–1027, 1993.

[6] Ron Kohavi and Mehran Sahami. Error-based and entropy-based discretization of continuous features. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD '96)*, pages 114–119, Portland, OR, 1996.

[7] Nir Friedman, Moises Goldszmidt, and Thomas J. Lee. Bayesian network classification with continuous attributes: Getting the best of both discretization and parametric fitting. In *Machine Learning: Proceedings of the 15th International Conference (ML '98)*, San Francisco, CA, 1998.

[8] Shyi-Ming Chen and Jeng-Yih Wang. Document retrieval using knowledge-based fuzzy information retrieval techniques. *Systems, Man and Cybernetics, IEEE Transactions on*, 25:793–803, 1995.

[9] Samuel S. Wilks. *Mathematical Statistics*. Wiley and Sons, New York, 1962.

[10] David Heckerman. A tutorial on learning with Bayesian networks. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 301–354. Kluwer Academic Publishers, Boston, 1998.

[11] M. N. Azaiez. *Perfect Aggregation in Reliability Models with BAyesian Updating*. PhD thesis, Department of Industrial Engineering, University of Wisconsin-Madison, Madison, Wisconsin, 1993.

[12] Y. Iwasa, S. Levin, and V. Andreasen. Aggregation in model ecosystem: Perfect aggregation. *Ecological Modeling*, 37:287–302, 1987.

[13] Tzu-Tsung Wong. *Perfect Aggregation in Dependent Bernoulli Systems with Bayesian Updating*. PhD thesis, Department of Industrial Engineering, University of Wisconsin-Madison, Madison, Wisconsin, 1998.

[14] Russell Almond. *Graphical Belief Modelling*. Chapman and Hall, New York, 1995.

[15] Bojan Cestnik and Ivan Bratko. On estimating probabilities in tree pruning. In *Machine Learning – EWSL-91, European Working Session on Learning*, pages 138–150. Springer-Verlag, Berlin, Germany, 1991.

[16] Chun-Nan Hsu, Hung-Ju Huang, and Tzu-Tsung Wong. Why discretization works for naive bayesian classifiers. In *Machine Learning: Proceedings of the 17th International Conference (ML 2000)*, San Francisco, CA, 2000.

[17] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.