A Comparative Study of Discretization Methods for Naive-Bayes Classifiers

Ying Yang[1] & Geoffrey I. Webb[2]

[1] School of Computing and Mathematics
Deakin University, VIC 3125, Australia
yyang@deakin.edu.au
[2] School of Computer Science and Software Engineering
Monash University, VIC 3800, Australia
Geoff.Webb@mail.csse.monash.edu.au

Abstract. Discretization is a popular approach to handling numeric attributes in machine learning. We argue that the requirements for effective discretization differ between naive-Bayes learning and many other learning algorithms. We evaluate the effectiveness with naive-Bayes classifiers of nine discretization methods, equal width discretization (EWD), equal frequency discretization (EFD), fuzzy discretization (FD), entropy minimization discretization (EMD), iterative discretization (ID), proportional k-interval discretization (PKID), lazy discretization (LD), non-disjoint discretization (NDD) and weighted proportional k-interval discretization (WPKID). It is found that in general naive-Bayes classifiers trained on data preprocessed by LD, NDD or WPKID achieve lower classification error than those trained on data preprocessed by the other discretization methods. But LD can not scale to large data. This study leads to a new discretization method, weighted non-disjoint discretization (WNDD) that combines WPKID and NDD's advantages. Our experiments show that among all the rival discretization methods, WNDD best helps naive-Bayes classifiers reduce average classification error.

1 Introduction

Classification tasks often involve numeric attributes. For naive-Bayes classifiers, numeric attributes are often preprocessed by discretization as the classification performance tends to be better when numeric attributes are discretized than when they are assumed to follow a normal distribution [9]. For each numeric attribute X, a categorical attribute X^* is created. Each value of X^* corresponds to an interval of values of X. X^* is used instead of X for training a classifier.

While many discretization methods have been employed for naive-Bayes classifiers, thorough comparisons of these methods are rarely carried out. Furthermore, many methods have only been tested on small datasets with hundreds of instances. Since large datasets with high dimensional attribute spaces and huge numbers of instances are increasingly used in real-world applications, a study of these methods' performance on large datasets is necessary and desirable [11, 27].

Nine discretization methods are included in this comparative study, each of which is either designed especially for naive-Bayes classifiers or is in practice often used for naive-Bayes classifiers. We seek answers to the following questions with experimental evidence:

- What are the strengths and weaknesses of the existing discretization methods applied to naive-Bayes classifiers?
- To what extent can each discretization method help reduce naive-Bayes classifiers' classification error?

Improving the performance of naive-Bayes classifiers is of particular significance as their efficiency and accuracy have led to widespread deployment.

As follows, Section 2 gives an overview of naive-Bayes classifiers. Section 3 introduces discretization for naive-Bayes classifiers. Section 4 describes each discretization method and discusses its suitability for naive-Bayes classifiers. Section 5 compares the complexities of these methods. Section 6 presents experimental results. Section 7 proposes weighted non-disjoint discretization (WNDD) inspired by this comparative study. Section 8 provides a conclusion.

2 Naive-Bayes Classifiers

In classification learning, each instance is described by a vector of attribute values and its class can take any value from some predefined set of values. Training data, a set of instances with known classes, are provided. A test instance is presented. The learner is asked to predict the class of the test instance according to the evidence provided by the training data. We define:

- C as a random variable denoting the class of an instance,
- $X < X_1, X_2, \cdots, X_k >$ as a vector of random variables denoting observed attribute values (an instance),
- c as a particular class,
- $x < x_1, x_2, \cdots, x_k >$ as a particular observed attribute value vector (a particular instance),
- $X = x$ as shorthand for $X_1 = x_1 \wedge X_2 = x_2 \wedge \cdots \wedge X_k = x_k$.

Expected classification error can be minimized by choosing $argmax_c(p(C = c \,|\, X = x))$ for each x. Bayes' theorem can be used to calculate the probability:

$$p(C = c \,|\, X = x) = \frac{p(C = c)\, p(X = x \,|\, C = c)}{p(X = x)}. \tag{1}$$

Since the denominator in (1) is invariant across classes, it does not affect the final choice and can be dropped, thus:

$$p(C = c \,|\, X = x) \propto p(C = c)\, p(X = x \,|\, C = c). \tag{2}$$

$p(C = c)$ and $p(X = x \,|\, C = c)$ need to be estimated from the training data. Unfortunately, since x is usually an unseen instance which does not appear in the

training data, it may not be possible to directly estimate $p(X = x \,|\, C = c)$. So a simplification is made: if attributes $X_1, X_2, \cdots, X_k$ are conditionally independent of each other given the class, then

$$\begin{aligned} p(X = x \,|\, C = c) &= p(\wedge X_i = x_i \,|\, C = c) \\ &= \prod p(X_i = x_i \,|\, C = c). \end{aligned} \tag{3}$$

Combining (2) and (3), one can further estimate the probability by:

$$p(C = c \,|\, X = x) \propto p(C = c) \prod p(X_i = x_i \,|\, C = c). \tag{4}$$

Classifiers using (4) are called naive-Bayes classifiers.

Naive-Bayes classifiers are simple, efficient and robust to noisy data. One limitation is that the attribute independence assumption in (3) is often violated in the real world. However, Domingos and Pazzani [8] suggest that this limitation has less impact than might be expected because classification under zero-one loss is only a function of the sign of the probability estimation; the classification accuracy can remain high even while the probability estimation is poor.

3 Discretization for Naive-Bayes Classifiers

An attribute is either categorical or numeric. Values of a categorical attribute are discrete. Values of a numeric attribute are either discrete or continuous [18].

$p(X_i = x_i \,|\, C = c)$ in (4) is modelled by a real number between 0 and 1, denoting the probability that the attribute X_i will take the particular value x_i when the class is c. This assumes that attribute values are discrete with a finite number, as it may not be possible to assign a probability to any single value of an attribute with an infinite number of values. Even for attributes that have a finite but large number of values, as there will be very few training instances for any one value, it is often advisable to aggregate a range of values into a single value for the purpose of estimating the probabilities in (4). In keeping with normal terminology of this research area, we call the conversion of a numeric attribute to a categorical one, *discretization*, irrespective of whether that numeric attribute is discrete or continuous.

A categorical attribute often takes a small number of values. So does the class label. Accordingly $p(C = c)$ and $p(X_i = x_i \,|\, C = c)$ can be estimated with reasonable accuracy from the frequency of instances with $C = c$ and the frequency of instances with $X_i = x_i \wedge C = c$ in the training data. In our experiment:

- The Laplace-estimate [6] was used to estimate $p(C = c)$: $\frac{n_c + k}{N + n \times k}$, where n_c is the number of instances satisfying $C = c$, N is the number of training instances, n is the number of classes and $k = 1$.
- The M-estimate [6] was used to estimate $p(X_i = x_i \,|\, C = c)$: $\frac{n_{ci} + m \times p}{n_c + m}$, where n_{ci} is the number of instances satisfying $X_i = x_i \wedge C = c$, n_c is the number of instances satisfying $C = c$, p is the prior probability $p(X_i = x_i)$ (estimated by the Laplace-estimate), and $m = 2$.

When a continuous numeric attribute X_i has a large or even an infinite number of values, as do many discrete numeric attributes, suppose S_i is the value space of X_i, for any particular $x_i \in S_i$, the probability $p(X_i = x_i)$ will be arbitrarily close to 0. The probability distribution of X_i is completely determined by a density function f which satisfies [30]:

1. $f(x_i) \geq 0, \forall x_i \in S_i$;
2. $\int_{S_i} f(X_i)\mathrm{d}X_i = 1$;
3. $\int_{a_i}^{b_i} f(X_i)\mathrm{d}X_i = p(a_i < X_i \leq b_i), \forall (a_i, b_i] \in S_i$.

$p(X_i = x_i \,|\, C = c)$ can be estimated from f [17]. But for real-world data, f is usually unknown. Under discretization, a categorical attribute X_i^* is formed for X_i. Each value x_i^* of X_i^* corresponds to an interval $(a_i, b_i]$ of X_i. If $x_i \in (a_i, b_i]$, $p(X_i = x_i|C = c)$ in (4) is estimated by

$$\begin{aligned} p(X_i = x_i|C = c) &\approx p(a < X_i \leq b \,|\, C = c) \\ &\approx p(X_i^* = x_i^*|C = c). \end{aligned} \tag{5}$$

Since X_i^* instead of X_i is used for training classifiers and $p(X_i^* = x_i^*|C = c)$ is estimated as for categorical attributes, probability estimation for X_i is not bounded by some specific distribution assumption. But the difference between $p(X_i = x_i|C = c)$ and $p(X_i^* = x_i^*|C = c)$ may cause information loss.

Although many discretization methods have been developed for learning other forms of classifiers, the requirements for effective discretization differ between naive-Bayes learning and those of most other learning contexts. Naive-Bayes classifiers are probabilistic, selecting the class with the highest probability given an instance. It is plausible that it is less important to form intervals dominated by a single class for naive-Bayes classifiers than for decision trees or decision rules. Thus discretization methods that pursue pure intervals (containing instances with the same class) [1, 5, 10, 11, 14, 15, 19, 29] might not suit naive-Bayes classifiers. Besides, naive-Bayes classifiers deem attributes conditionally independent of each other and do not use attribute combinations as predictors. There is no need to calculate the joint probabilities of multiple attribute-values. Thus discretization methods that seek to capture inter-dependencies among attributes [3, 7, 13, 22–24, 26, 31] might be less applicable to naive-Bayes classifiers.

Instead, for naive-Bayes classifiers it is required that discretization should result in accurate estimation of $p(X_i = x_i|C = c)$. It is also required that discretization should be efficient in order to maintain naive-Bayes classifiers' desirable computational efficiency.

4 Discretization Methods

When discretizing a numeric attribute X_i, suppose there are n training instances for which the value of X_i is known, the minimum and maximum value are v_{min} and v_{max} respectively, each discretization method first sorts the values into ascending order. The methods then differ as follows.

4.1 Equal Width Discretization (EWD)

EWD [5, 9, 19] divides the number line between v_{min} and v_{max} into k intervals of equal width. Thus the intervals have width $w = (v_{max} - v_{min})/k$ and the cut points are at $v_{min} + w, v_{min} + 2w, \cdots, v_{min} + (k-1)w$. k is a user predefined parameter and is set as 10 in our experiments.

4.2 Equal Frequency Discretization (EFD)

EFD [5, 9, 19] divides the sorted values into k intervals so that each interval contains approximately the same number of training instances. Thus each interval contains n/k (possibly duplicated) adjacent values. k is a user predefined parameter and is set as 10 in our experiments.

Both EWD and EFD potentially suffer much attribute information loss since k is determined without reference to the properties of the training data. But although they may be deemed simplistic, both methods are often used and work surprisingly well for naive-Bayes classifiers. One reason suggested is that discretization usually assumes that discretized attributes have Dirichlet priors, and 'Perfect Aggregation' of Dirichlets can ensure that naive-Bayes with discretization appropriately approximates the distribution of a numeric attribute [16].

4.3 Fuzzy Discretization (FD)

There are three versions of fuzzy discretization proposed by Kononenko for naive-Bayes classifiers [20, 21]. They differ in how the estimation of $p(a_i < X_i \leq b_i \,|\, C = c)$ in (5) is obtained. Because space limits, we present here only the version that, according to our experiments, best reduces the classification error.

FD initially forms k equal-width intervals $(a_i, b_i]$ $(1 \leq i \leq k)$ using EWD. Then FD estimates $p(a_i < X_i \leq b_i \,|\, C = c)$ from all training instances rather than from instances that have value of X_i in $(a_i, b_i]$. The influence of a training instance with value v of X_i on $(a_i, b_i]$ is assumed to be normally distributed with the mean value equal to v and is proportional to $P(v, \sigma, i) = \int_{a_i}^{b_i} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-v}{\sigma})^2} \mathrm{d}x$. σ is a parameter to the algorithm and is used to control the 'fuzziness' of the interval bounds. σ is set equal to $0.7 \times \frac{v_{max} - v_{min}}{k}$ [1]. Suppose there are n_c training instances with known value for X_i and with class c, each with influence $P(v_j, \sigma, i)$ on $(a_i, b_i]$ $(j = 1, \cdots, n_c)$:

$$p(a_i < X_i \leq b_i \,|\, C = c) = \frac{p(a_i < X_i \leq b_i \wedge C = c)}{p(C = c)} \approx \frac{\frac{\sum_{j=1}^{n_c} P(v_j, \sigma, i)}{n}}{p(C = c)}. \quad (6)$$

The idea behind fuzzy discretization is that small variation of the value of a numeric attribute should have small effects on the attribute's probabilities, whereas under non-fuzzy discretization, a slight difference between two values,

[1] This setting of σ is chosen because it has been shown to achieve the best results in Kononenko's experiments [20].

one above and one below the cut point can have drastic effects on the estimated probabilities. But when the training instances' influence on each interval does not follow the normal distribution, FD's performance can degrade. The number of initial intervals k is a predefined parameter and is set as 10 in our experiments.

4.4 Entropy Minimization Discretization (EMD)

EMD [10] evaluates as a candidate cut point the midpoint between each successive pair of the sorted values. For evaluating each candidate cut point, the data are discretized into two intervals and the resulting class information entropy is calculated. A binary discretization is determined by selecting the cut point for which the entropy is minimal amongst all candidates. The binary discretization is applied recursively, always selecting the best cut point. A minimum description length criterion (MDL) is applied to decide when to stop discretization.

Although it is often used with naive-Bayes classifiers, EMD was developed in the context of top-down induction of decision trees. It uses MDL as the termination condition to form categorical attributes with few values. For decision tree learning, it is important to minimize the number of values of an attribute, so as to avoid the fragmentation problem [28]. If an attribute has many values, a split on this attribute will result in many branches, each of which receives relatively few training instances, making it difficult to select appropriate subsequent tests. Naive-Bayes learning considers attributes independent of one another given the class, and hence is not subject to the fragmentation problem if there are many values for an attribute. So EMD's bias towards forming a small number of intervals may not be so well justified for naive-Bayes classifiers as for decision trees.

Another unsuitability of EMD for naive-Bayes classifiers is that EMD discretizes a numeric attribute by calculating the class information entropy as if the naive-Bayes classifiers only use that *single* attribute after discretization. Thus, EMD makes a form of attribute independence assumption when discretizing. There is a risk that this might reinforce the attribute independence assumption inherent in naive-Bayes classifiers, further reducing their capability to accurately classify in the context of violation of the attribute independence assumption.

4.5 Iterative Discretization (ID)

ID [25] initially forms a set of intervals using EWD or MED, and then iteratively adjusts the intervals to minimize naive-Bayes classifiers' classification error on the training data. It defines two operators: merge two contiguous intervals and split an interval into two intervals by introducing a new cut point that is midway between each pair of contiguous values in that interval. In each loop of the iteration, for each numeric attribute, ID applies all operators in all possible ways to the current set of intervals and estimates the classification error of each adjustment using leave-one-out cross-validation. The adjustment with the lowest error is retained. The loop stops when no adjustment further reduces the error.

A disadvantage of ID results from its iterative nature. ID discretizes a numeric attribute with respect to error on the training data. When the training data are large, the possible adjustments by applying the two operators will tend to be immense. Consequently the repetitions of the leave-one-out cross validation will be prohibitive so that ID is infeasible in terms of time consumption. Since our experiments include large datasets, we hereby only introduce ID for the integrality of our study, without implementing it.

4.6 Proportional k-Interval Discretization (PKID)

PKID [33] adjusts discretization bias and variance by tuning the interval size and number, and further adjusts the naive-Bayes' probability estimation bias and variance to achieve lower classification error. The idea behind PKID is that discretization bias and variance relate to interval size and interval number. The larger the interval size (the smaller the interval number), the lower the variance but the higher the bias. In the converse, the smaller the interval size (the larger the interval number), the lower the bias but the higher the variance. Lower learning error can be achieved by tuning the interval size and number to find a good trade-off between the bias and variance. Suppose the desired interval size is s and the desired interval number is t, PKID employs (7) to calculate s and t:

$$\begin{aligned} s \times t &= n \\ s &= t. \end{aligned} \tag{7}$$

PKID discretizes the sorted values into intervals with size s. Thus PKID gives equal weight to discretization bias reduction and variance reduction by setting the interval size equal to the interval number ($s = t \approx \sqrt{n}$). Furthermore, PKID sets both interval size and number proportional to the training data size. As the quantity of the training data increases, both discretization bias and variance can decrease. This means that PKID has greater capacity to take advantage of the additional information inherent in large volumes of training data.

Previous experiments [33] showed that PKID significantly reduced classification error for larger datasets. But PKID was sub-optimal for smaller datasets[2]. This might be because when n is small PKID produces a number of intervals with small size which might not offer enough data for reliable probability estimation, thus resulting in high variance and inferior performance of naive-Bayes classifiers.

4.7 Lazy Discretization (LD)

LD [16] defers discretization until classification time estimating $p(X_i = x_i \,|\, C = c)$ in (4) for each attribute of each test instance. It waits until a test instance is

[2] There is no strict definition of 'smaller' and 'larger' datasets. This research deems datasets with size no larger than 1000 as 'smaller' datasets, otherwise as 'larger' datasets.

presented to determine the cut points for X_i. For the value of X_i from the test instance, it selects a pair of cut points such that the value is in the middle of its corresponding interval whose size is the same as created by EFD with $k = 10$. However 10 is an arbitrary number which has never been justified as superior to any other value.

LD tends to have high memory and computational requirements because of its lazy methodology. Eager approaches carry out discretization prior to the naive-Bayes learning. They only keep training instances to estimate the probabilities in (4) during training time and discard them during classification time. In contrast, LD needs to keep training instances for use during classification time. This demands high memory when the training data are large. Besides, where a large number of instances need to be classified, LD will incur large computational overheads since it does not estimate probabilities until classification time.

Although LD achieves comparable performance to EFD and EMD [16], the high memory and computational overheads might impede it from feasible implementation for classification tasks with large training or test data. In our experiments, we implement LD with the interval size equal to that created by PKID, as this has been shown to outperform the original implementation [16, 34].

4.8 Non-Disjoint Discretization (NDD)

NDD [34] forms overlapping intervals for X_i, always locating a value x_i toward the middle of its corresponding interval $(a_i, b_i]$. The idea behind NDD is that when substituting $(a_i, b_i]$ for x_i, there is more distinguishing information about x_i and thus the probability estimation in (5) is more reliable if x_i is in the middle of $(a_i, b_i]$ than if x_i is close to either boundary of $(a_i, b_i]$. LD embodies this idea in the context of lazy discretization. NDD employs an eager approach, performing discretization prior to the naive-Bayes learning.

NDD bases its interval size strategy on PKID. Figure 1 illustrates the procedure. Given s and t calculated as in (7), NDD identifies among the sorted values t' *atomic interval*s, $(a'_1, b'_1], (a'_2, b'_2], ..., (a'_{t'}, b'_{t'}]$, each with size equal to s', so that[3]

$$s' = \frac{s}{3}$$
$$s' \times t' = n. \tag{8}$$

One interval is formed for each set of three consecutive *atomic interval*s, such that the kth ($1 \leq k \leq t' - 2$) interval $(a_k, b_k]$ satisfies $a_k = a'_k$ and $b_k = b'_{k+2}$. Each value v is assigned to interval $(a'_{i-1}, b'_{i+1}]$ where i is the index of the *atomic interval* $(a'_i, b'_i]$ such that $a'_i < v \leq b'_i$, except when $i = 1$ in which case v is assigned to interval $(a'_1, b'_3]$ and when $i = t'$ in which case v is assigned to interval $(a'_{t'-2}, b'_{t'}]$. As a result, except in the case of falling into the first or the last *atomic interval*, x_i is always toward the middle of $(a_i, b_i]$.

[3] Theoretically any odd number k besides 3 is acceptable in (8) as long as the same number k of *atomic interval*s are grouped together later for the probability estimation. For simplicity, we take $k = 3$ for demonstration.

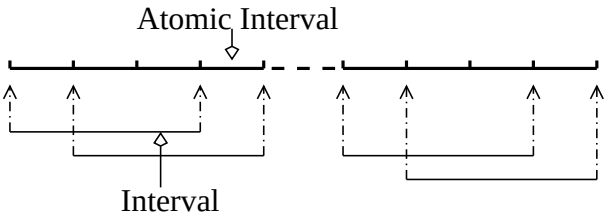

Fig. 1. *Atomic Interval*s Compose Actual Intervals

4.9 Weighted Proportional k-Interval Discretization (WPKID)

WPKID [35] is an improved version of PKID. It is credible that for smaller datasets, variance reduction can contribute more to lower naive-Bayes learning error than bias reduction [12]. Thus fewer intervals each containing more instances would be of greater utility. Accordingly WPKID weighs discretization variance reduction more than bias reduction by setting a minimum interval size to make more reliable the probability estimation. As the training data increase, both the interval size *above* the minimum and the interval number increase.

Given the same definitions s and t as in (7), suppose the minimum interval size is $\mathbf{m}$, WPKID employs 9 to calculate s and t:

$$\begin{aligned} s \times t &= n \\ s - \mathbf{m} &= t \\ \mathbf{m} &= 30. \end{aligned} \tag{9}$$

$\mathbf{m}$ is set to 30 as it is commonly held as the minimum sample from which one should draw statistical inferences. This strategy should mitigate PKID's disadvantage on smaller datasets by establishing a suitable bias and variance trade-off, while it retains PKID's advantage on larger datasets by allowing additional training data to be used to reduce both bias and variance.

5 Algorithm Complexity Comparison

Suppose the number of training instances[4] and classes are n and m respectively.

- EWD, EFD, FD, PKID, WPKID and NDD are dominated by sorting. Their complexities are of order $O(n \log n)$.
- EMD does sorting first, an operation of complexity $O(n \log n)$. It then goes through all the training instances a maximum of $\log n$ times, recursively applying 'binary division' to find out at most $n - 1$ cut points. Each time, it will estimate $n - 1$ candidate cut points. For each candidate point, probabilities of each of m classes are estimated. The complexity of that operation is $O(mn \log n)$, which dominates the complexity of the sorting, resulting in complexity of order $O(mn \log n)$.

[4] Consider only instances with known value of the numeric attribute to be discretized.

– ID's operators have $O(n)$ possible ways to adjust in each iteration. For each adjustment, the leave-one-out cross validation has complexity of order $O(nmv)$ where v is the number of the attributes. If the iteration repeats u times until there is no further error reduction of leave-one-out cross-validation, the complexity of ID is of order $O(n^2mvu)$.
– LD performs discretization separately for each test instance and hence its complexity is $O(nl)$, where l is the number of test instances.

Thus EWD, EFD, FD, PKID, WPKID and NDD have low complexity. EMD's complexity is higher. LD tends to have high complexity when the testing data are large. ID's complexity is prohibitively high when the training data are large.

6 Experimental Validation

6.1 Experimental Setting

We run experiments on 35 natural datasets from UCI machine learning repository [4] and KDD archive [2]. These datasets vary extensively in the number of instances and the dimension of the attribute space. Table 1 describes each dataset, including the number of instances (Size), numeric attributes (Num.), categorical attributes (Cat.) and classes (Class).

For each dataset, we implement naive-Bayes learning by conducting a 10-trial, 3-fold cross validation. For each fold, the training data are separately discretized by EWD, EFD, FD, MED, PKID, LD, NDD and WPKID. The intervals so formed are separately applied to the test data. The experimental results are recorded as *average classification error* that, listed in Table 1, is the percentage of incorrect predictions of naive-Bayes classifiers in the test across trials.

6.2 Experimental Statistics

Three statistics are employed to evaluate the experimental results.

Mean error. This is the mean of errors across all datasets. It provides a gross indication of relative performance. It is debatable whether errors in different datasets are commensurable, and hence whether averaging errors across datasets is very meaningful. Nonetheless, a low average error is indicative of a tendency toward low errors for individual datasets.

Geometric mean error ratio. This method has been explained by Webb [32]. It allows for the relative difficulty of error reduction in different datasets and can be more reliable than the mean ratio of errors across datasets.

Win/lose/tie record. The three values are respectively the number of datasets for which a method obtains lower, higher or equal classification error, compared with an alternative method. A one-tailed sign test can be applied to each record. If the test result is significantly low (here we use the 0.05 critical level), it is reasonable to conclude that the outcome is unlikely to be obtained by chance and hence the record of wins to losses represents a systematic underlying advantage to the winning method with respect to the type of datasets studied.

Table 1. Experimental Datasets and Classification Error

Dataset	Size	Num.	Cat.	Class	Classification Error(%)								
					EWD	EFD	FD	EMD	PKID	LD	NDD	WPKID	WNDD
Labor-Negotiations	57	8	8	2	12.3	8.9	12.8	9.5	7.2	7.7	7.0	8.6	5.3
Echocardiogram	74	5	1	2	29.6	29.2	27.7	23.8	25.3	26.2	26.6	25.7	24.2
Pittsburgh-Bridges-Material	106	3	8	3	12.9	12.1	10.5	12.6	13.0	12.3	13.1	11.9	10.8
Iris	150	4	0	3	5.7	7.5	5.3	6.8	7.5	6.7	7.2	6.9	7.3
Hepatitis	155	6	13	2	14.9	14.7	13.4	14.5	14.6	14.2	14.4	15.9	15.6
Wine-Recognition	178	13	0	3	3.3	2.1	3.3	2.6	2.2	3.8	3.3	2.0	1.9
Flag-Landmass	194	10	18	6	30.8	30.5	32.0	29.9	30.7	30.9	30.5	29.0	29.0
Sonar	208	60	0	2	26.9	25.2	26.8	26.3	25.7	27.3	26.9	23.7	22.8
Glass-Identification	214	9	0	3	41.9	39.4	44.5	36.8	40.4	22.2	38.8	38.9	35.3
Heart-Disease(Cleveland)	270	7	6	2	18.3	17.1	16.3	17.5	17.5	17.8	18.6	16.7	16.7
Haberman	306	3	0	2	26.5	27.1	25.1	26.5	27.7	27.5	27.8	25.8	26.1
Ecoli	336	5	2	8	18.5	19.0	16.0	17.9	19.0	20.2	20.2	17.6	17.4
Liver-Disorders	345	6	0	2	37.1	37.1	37.9	37.4	38.0	38.0	37.7	35.5	35.9
Ionosphere	351	34	0	2	9.4	10.2	8.5	11.1	10.6	11.3	10.2	10.3	10.6
Dermatology	366	1	33	6	2.3	2.2	1.9	2.0	2.2	2.3	2.4	1.9	1.9
Horse-Colic	368	8	13	2	20.5	20.9	20.7	20.7	20.9	19.7	20.0	20.7	20.6
Credit-Screening(Australia)	690	6	9	2	15.6	14.5	15.2	14.5	14.2	14.7	14.4	14.3	14.1
Breast-Cancer(Wisconsin)	699	9	0	2	2.5	2.6	2.8	2.7	2.7	2.7	2.6	2.7	2.7
Pima-Indians-Diabetes	768	8	0	2	24.9	25.9	24.8	26.0	26.3	26.4	25.8	25.5	25.4
Vehicle	846	18	0	4	38.7	40.5	42.4	38.9	38.2	38.7	38.5	38.2	38.8
Annealing	898	6	32	6	3.5	2.3	3.9	1.9	2.2	1.6	1.8	2.2	2.3
Vowel-Context	990	10	1	11	35.1	38.4	38.0	41.4	43.0	41.1	43.7	39.2	37.2
German	1000	7	13	2	25.4	25.4	25.2	25.1	25.5	25.0	25.4	25.4	25.1
Multiple-Features	2000	3	3	10	30.9	31.9	30.8	32.6	31.5	31.2	31.6	31.4	30.9
Hypothyroid	3163	7	18	2	3.5	2.8	2.6	1.7	1.8	1.7	1.7	2.1	1.7
Satimage	6435	36	0	6	18.8	18.9	20.1	18.1	17.8	17.5	17.5	17.7	17.6
Musk	6598	166	0	2	13.7	19.2	21.2	9.4	8.3	7.8	7.7	8.5	8.2
Pioneer-MobileRobot	9150	29	7	57	9.0	10.8	18.2	14.8	1.7	1.7	1.6	1.8	2.0
Handwritten-Digits	10992	16	0	10	12.5	13.2	13.2	13.5	12.0	12.1	12.1	12.2	12.0
Australian-Sign-Language	12546	8	0	3	38.3	38.2	38.7	36.5	35.8	35.8	35.8	36.0	35.8
Letter-Recognition	20000	16	0	26	29.5	30.7	34.7	30.4	25.8	25.5	25.6	25.7	25.6
Adult	48842	6	8	2	18.2	19.2	18.5	17.2	17.1	17.1	17.0	17.0	17.1
Ipums-la-99	88443	20	40	13	20.2	20.5	32.0	20.1	19.9	19.1	18.6	19.9	18.6
Census-Income	299285	8	33	2	24.5	24.5	24.7	23.6	23.3	23.6	23.3	23.3	23.3
Forest-Covertype	581012	10	44	7	32.4	32.9	32.2	32.1	31.7	-	31.4	31.7	31.4
ME	-	-	-	-	20.1	19.9	20.9	19.5	19.1	18.6	19.1	18.7	18.2
GM	-	-	-	-	1.15	1.14	1.19	1.09	1.02	1.00	1.02	1.00	0.97

6.3 Experimental Results Analysis

WPKID is the latest discretization technique proposed for naive-Bayes classifiers. It was claimed to outperform previous techniques EFD, FD, EMD and PKID [35]. No comparison has yet been done among EWD, LD, NDD and WPKID. In our analysis, we take WPKID as a benchmark, against which we study the performance of the other methods. The dataset Forest-Covertype is excluded from the calculations of the mean error, the geometric mean error ratio and the win/lose/tie records involving LD because it could not be completed by LD during a tolerable time.

- The 'ME' row of Table 1 presents the mean error of each method. LD achieves the lowest mean error with WPKID achieving a very similar score.
- The 'GE' row of Table 1 presents the geometric mean ratio of each method against WPKID. All results except for LD are larger than 1. This suggests

that WPKID and LD enjoy an advantage in terms of error reduction over the type of datasets studied in this research.

– With respect to the win/lose/tie records of the other methods against WPKID, the results are listed in Table 2. The sign test shows that EWD, EFD, FD, EMD and PKID are worse than WPKID at error reduction with frequency significant at the 0.05 level. LD and NDD have competitive performance compared with WPKID.

Table 2. Win/Lose/Tie Records of Alternative Methods against WPKID

-	EWD	EFD	FD	EMD	PKID	LD	NDD	WPKID
Win	8	4	11	7	9	16	16	-
Lose	26	30	22	26	20	17	16	-
Tie	1	1	2	2	6	1	3	-
Sign Test	< 0.01	< 0.01	0.04	< 0.01	0.03	0.50	0.60	-

– NDD and LD have similar error performance since they are similar in terms of locating an attribute value toward the middle of its discretized interval. The win/lose/tie record of NDD against LD is 15/14/5, resulting in sign test equal to 0.50. But from the view of computation time, NDD is overwhelmingly superior to LD. Table 3 lists the computation time of training and testing a naive-Bayes classifier on data preprocessed by NDD and LD respectively in one fold out of a 3-fold cross validation for the four largest datasets[5]. NDD is much faster than the LD.

Table 3. Computation Time for One Fold (Seconds)

-	Adult	Ipums.la.99	Census-Income	Forest-Covertype
NDD	0.7	13	10	60
LD	6025	691089	510859	> 864000

– Another interesting comparison is between EWD and EFD. It was said that EWD is vulnerable to outliers that may drastically skew the value range [9]. But according to our experiments, the win/lose/tie record of EWD against EFD is 18/14/3, which means that EWD performs at least as well as EFD if not better. This might be because naive-Bayes classifiers take all attributes

[5] For Forest-Covertype, LD was not able to obtain the classification result even after running for 864000 seconds. Allowing for the feasibility for real-world classification tasks, it was meaningless to keep LD running. So we stopped its process, resulting in no precise record for the running time.

into account simultaneously. Hence, the impact of a 'wrong' discretization of one attribute can be absorbed by other attributes under 'zero-one' loss [16]. Another observation is that the advantage of EWD over EFD is more apparent with training data increasing. The reason might be that the more training data available, the less the impact of an outlier.

7 Further Discussion

Our experiments show that LD, NDD and WPKID are better than the alternatives at reducing naive-Bayes classifiers' classification error. But LD is infeasible in the context of large datasets. NDD and WPKID's good performance can be attributed to the fact that they both focus on accurately estimating naive-Bayes' probabilities. NDD can retain more distinguishing information for a value to be discretized, while WPKID can properly adjust learning variance and bias.

A consequent interesting question is that what will happen if we combine NDD and WPKID together? Is it possible to obtain a discretization method even better reducing naive-Bayes classifiers' classification error? We name this new method weighted non-disjoint discretization (WNDD). WNDD follows NDD in terms of combining three *atomic interval*s into one interval. But WNDD has its interval size equal as produced by WPKID, while NDD has its interval size equal as produced by PKID.

We implement WNDD the same way as for the other discretization methods and record the resulting classification error in column 'WNDD' in Table 1. The experiments show that WNDD achieves the lowest ME and GM. The win/lose/tie records of previous methods against WNDD are listed in Table 4. WNDD achieves lower classification error than all other methods except LD and NDD with frequency significant at the 0.05 level. Although not statistically significant, WNDD delivers lower classification error more often than not in comparison with LD and NDD. WNDD also overcomes the computational limitation of LD since it has complexity as low as NDD.

Table 4. Win/Lose/Tie Records of Alternative Methods against WNDD

-	EWD	EFD	FD	EMD	PKID	WPKID	LD	NDD	WNDD
Win	8	3	9	4	4	8	11	11	-
Lose	26	31	25	28	25	22	19	18	-
Tie	1	1	1	3	6	5	4	6	-
Sign Test	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	0.10	0.13	-

8 Conclusion

This is an evaluation and comparison of discretization methods for naive-Bayes classifiers. We have discovered that in terms of reducing classification error, LD,

NDD and WPKID perform best. But LD's lazy methodology impedes it from scaling to large data. This analysis leads to a new discretization method WNDD which combines NDD and WPKID's advantages. Our experiments demonstrate that WNDD performs at least as well as the best existing techniques at minimizing classification error. This outstanding performance is achieved without the computational overheads that hamper the application of lazy discretization.

References

1. An, A., and Cercone, N. Discretization of continuous attributes for learning classification rules. In *Proc. Third Pacific-Asia Conf. on Methodologies for Knowledge Discovery and Data Mining* (1999), pp. 509–514.
2. Bay, S. D. The UCI KDD archive [http://kdd.ics.uci.edu], 1999. Department of Information and Computer Science, University of California, Irvine.
3. Bay, S. D. Multivariate discretization of continuous variables for set mining. In *Proc. Sixth ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining* (2000), pp. 315–319.
4. Blake, C. L., and Merz, C. J. UCI repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/mlrepository.html], 1998. Department of Information and Computer Science, University of California, Irvine.
5. Catlett, J. On changing continuous attributes into ordered discrete attributes. In *Proc. European Working Session on Learning* (1991), pp. 164–178.
6. Cestnik, B. Estimating probabilities: A crucial task in machine learning. In *Proc. European Conf. on Artificial Intelligence* (1990), pp. 147–149.
7. Chmielewski, M. R., and Grzymala-Busse, J. W. Global discretization of continuous attributes as preprocessing for machine learning. *International Journal of Approximate Reasoning 15* (1996), 319–331.
8. Domingos, P., and Pazzani, M. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning 29* (1997), 103–130.
9. Dougherty, J., Kohavi, R., and Sahami, M. Supervised and unsupervised discretization of continuous features. In *Proc. Twelfth International Conf. on Machine Learning* (1995), pp. 194–202.
10. Fayyad, U. M., and Irani, K. B. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. Thirteenth International Joint Conf. on Artificial Intelligence* (1993), pp. 1022–1027.
11. Freitas, A. A., and Lavington, S. H. Speeding up knowledge discovery in large relational databases by means of a new discretization algorithm. In *Advances in Databases, Proc. Fourteenth British National Conferenc on Databases* (1996), pp. 124–133.
12. Friedman, J. H. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery 1*, 1 (1997), 55–77.
13. Gama, J., Torgo, L., and Soares, C. Dynamic discretization of continuous attributes. In *Proc. Sixth Ibero-American Conf. on AI* (1998), pp. 160–169.
14. Ho, K. M., and Scott, P. D. Zeta: A global method for discretization of continuous variables. In *Proc. Third International Conf. on Knowledge Discovery and Data Mining* (1997), pp. 191–194.
15. Holte, R. C. Very simple classification rules perform well on most commonly used datasets. *Machine Learning 11* (1993), 63–91.

16. Hsu, C. N., Huang, H. J., and Wong, T. T. Why discretization works for naive Bayesian classifiers. In *Proc. Seventeenth International Conf. on Machine Learning* (2000), pp. 309–406.
17. John, G. H., and Langley, P. Estimating continuous distributions in Bayesian classifiers. In *Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence* (1995), pp. 338–345.
18. Johnson, R., and Bhattacharyya, G. *Statistics: Principles and Methods*. John Wiley & Sons Publisher, 1985.
19. Kerber, R. Chimerge: Discretization for numeric attributes. In *National Conf. on Artificial Intelligence* (1992), AAAI Press, pp. 123–128.
20. Kononenko, I. Naive Bayesian classifier and continuous attributes. *Informatica 16*, 1 (1992), 1–8.
21. Kononenko, I. Inductive and Bayesian learning in medical diagnosis. *Applied Artificial Intelligence 7* (1993), 317–337.
22. Kwedlo, W., and Kretowski, M. An evolutionary algorithm using multivariate discretization for decision rule induction. In *Proc. European Conf. on Principles of Data Mining and Knowledge Discovery* (1999), pp. 392–397.
23. Ludl, M.-C., and Widmer, G. Relative unsupervised discretization for association rule mining. In *Proc. Fourth European Conf. on Principles and Practice of Knowledge Discovery in Databases* (2000).
24. Monti, S., and Cooper, G. A multivariate discretization method for learning bayesian networks from mixed data. In *Proc. Fourteenth Conf. of Uncertainty in AI* (1998), pp. 404–413.
25. Pazzani, M. J. An iterative improvement approach for the discretization of numeric attributes in Bayesian classifiers. In *Proc. First International Conf. on Knowledge Discovery and Data Mining* (1995).
26. Perner, P., and Trautzsch, S. Multi-interval discretization methods for decision tree learning. In *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR '98 and SPR '98* (1998), pp. 475–482.
27. Provost, F. J., and Aronis, J. M. Scaling up machine learning with massive parallelism. *Machine Learning 23* (1996).
28. Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.
29. Richeldi, M., and Rossotto, M. Class-driven statistical discretization of continuous attributes (extended abstract). In *European Conf. on Machine Learning* (335-338, 1995), Springer.
30. Scheaffer, R. L., and McClave, J. T. *Probability and Statistics for Engineers*, fourth ed. Duxbury Press, 1995.
31. Wang, K., and Liu, B. Concurrent discretization of multiple attributes. In *The Pacific Rim International Conf. on Artificial Intelligence* (1998), pp. 250–259.
32. Webb, G. I. Multiboosting: A technique for combining boosting and wagging. *Machine Learning 40*, 2 (2000), 159–196.
33. Yang, Y., and Webb, G. I. Proportional k-interval discretization for naive-Bayes classifiers. In *Proc. Twelfth European Conf. on Machine Learning* (2001), pp. 564–575.
34. Yang, Y., and Webb, G. I. Non-disjoint discretization for naive-Bayes classifiers. In *Proc. Nineteenth International Conf. on Machine Learning* (2002), pp. 666–673.
35. Yang, Y., and Webb, G. I. Weighted proportional k-interval discretization for naive-Bayes classifiers. In *Submitted to The 2002 IEEE International Conf. on Data Mining* (2002).