Non-Disjoint Discretization for Naive-Bayes Classifiers

Ying Yang YYANG@DEAKIN.EDU.AU
Geoffrey I. Webb WEBB@DEAKIN.EDU.AU
School of Computing and Mathematics, Deakin University, Vic3217, Australia

Abstract

Previous discretization techniques have discretized numeric attributes into disjoint intervals. We argue that this is neither necessary nor appropriate for naive-Bayes classifiers. The analysis leads to a new discretization method, Non-Disjoint Discretization (NDD). NDD forms overlapping intervals for a numeric attribute, always locating a value toward the middle of an interval to obtain more reliable probability estimation. It also adjusts the number and size of discretized intervals to the number of training instances, seeking an appropriate trade-off between bias and variance of probability estimation. We justify NDD in theory and test it on a wide cross-section of datasets. Our experimental results suggest that for naive-Bayes classifiers, NDD works better than alternative discretization approaches.

1. Introduction

The speed of naive-Bayes classifiers has seen them deployed in numerous classification tasks. Many of those tasks involve numeric attributes. For naive-Bayes classifiers, numeric attributes are often discretized (Dougherty et al., 1995). For each numeric attribute A, a categorical attribute A^* is created. Each value of A^* corresponds to an interval of A. When training a classifier, the learning process uses A^* instead of A.

For naive-Bayes classifiers, one common discretization approach is fixed k-interval discretization (FKID) (Catlett, 1991). Another popular one is Fayyad and Irani's entropy minimization heuristic discretization (FID) (Fayyad & Irani, 1993). Two recent alternatives are lazy discretization (LD) (Hsu et al., 2000) and proportional k-interval discretization (PKID) (Yang & Webb, 2001).

In this paper, we first argue that those approaches are inappropriate for naive-Bayes classifiers. We then propose a new discretization scheme, non-disjoint discretization (NDD). Each of FKID, FID and PKID partitions the range of a numeric attribute's values offered by the training data into *disjoint* intervals. Many learning algorithms require values of an attribute to be disjoint, the set of instances covered by one value can not overlap that covered by another value. Naive-Bayes classifiers do not have that requirement. Only one value of each attribute which applies to an instance is used for classifying that instance. The remaining values can be ignored. During classification, each numeric value's conditional probability given a class is estimated from the training data by the frequency of the co-ocurrence of the value's discretized interval and the class, and the frequency of the class. For values near either boundary of the interval, the estimation might not be so reliable as for values in the middle of the interval. Accordingly, instead of forming *disjoint* intervals, we form *overlapping* intervals to always locate a numeric value toward the middle of the interval to which it belongs. LD also settles a value in the middle of an interval. But because of its lazy methodology, LD has low computational efficiency. In contrast, NDD is eager in that it creates all intervals during training time, resulting in more efficient computation.

To evaluate NDD, we separately implement FKID, FID, LD, PKID, and NDD to train naive-Bayes classifiers. We compare the classification errors of the resulting classifiers. Our hypothesis is that naive-Bayes classifiers trained on data preprocessed by NDD are able to achieve lower classification error, compared with those trained on data preprocessed by the alternatives.

As follows, Section 2 introduces naive-Bayes classifiers. Section 3 describes FKID, FID, LD, and PKID. Section 4 discusses NDD. Section 5 compares the computational complexities. Section 6 presents experimental results. Section 7 provides a conclusion.

2. Naive-Bayes Classifiers

Naive-Bayes classifiers are simple, efficient and robust to noisy data. One limitation, however, is that naive-

Bayes classifiers utilize an assumption that attributes are conditionally independent of each other given a class. Although this assumption is often violated in the real world, the classification performance of naive-Bayes classifiers is surprisingly good compared with other more complex classifiers. According to Domingos and Pazzani (1997), this is explained by the fact that classification estimation 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.

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. A set of instances with their classes, the training data, is provided. A test instance is presented. The learner is asked to predict its class 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 the observed attribute values (an instance),
- c as a particular class label,
- $x < x_1, x_2, \cdots, x_k >$ 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:

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

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

$$p(C = c \,|\, X = x) \propto p(C = c)\, p(X = x \,|\, C = c). \quad (2)$$

$p(C = c)$ 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:

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

Combining (2) and (3), one can further estimate the most probable class by using:

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

Classifiers using (4) are called naive-Bayes classifiers. The independence assumption embodied in (3) makes the computation of naive-Bayes classifiers more efficient than the exponential complexity of non-naive Bayes approaches because it does not use attribute combinations as predictors (Yang & Liu, 1999).

3. Discretize Numeric Attributes

An attribute is either categorical or numeric. Values of a categorical attribute are discrete. Values of a numeric attribute are either discrete or continuous. (Johnson & Bhattacharyya, 1985)

$p(X_i = x_i \,|\, C = c)$ in (4) is modeled by a single 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 discrete 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 attribute, *discretization*, irrespective of whether this numeric attribute is discrete or continuous.

A categorical attribute often takes a small number of values. So does the class label. In consequence, $p(C = c)$ 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,

Laplace-estimate (Cestnik, 1990) is used to estimate $p(C = c)$: $\frac{n_c + k}{N + n * k}$, 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 $k = 1$.

M-estimate (Cestnik, 1990) is used to estimate $p(X_i = x_i \,|\, C = c)$: $\frac{n_{ci} + mp}{n_c + m}$, 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 $p(X_i = x_i)$ (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 $x_i \in S_i$, $p(X_i = x_i)$ will be arbitrarily close to 0. The probability distribution of X_i is completely determined by a density function f which satisfies (Scheaffer & McClave, 1995):

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 (John & Langley, 1995). 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. X_i^* instead of X_i is employed for training classifiers. $p(X_i^* = x_i^* | C = c)$ is estimated as for categorical attributes. In consequence, 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.

3.1 Fixed k-Interval Discretization (FKID)

FKID (Catlett, 1991) divides the sorted values of a numeric attribute into k intervals, where (given n observed instances) each interval contains n/k (possibly duplicated) adjacent values. k is determined without reference to the properties of the training data[1]. This method ignores relationships among different values, thus potentially suffers much attribute information loss. But although it may be deemed simplistic, this technique works surprisingly well for naive-Bayes classifiers. One suggestion is that discretization approaches usually assume 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 (Hsu et al., 2000).

3.2 Fayyad and Irani's Entropy Minimization Heuristic Discretization (FID)

For a numeric attribute, FID (Fayyad & Irani, 1993) evaluates as a candidate cut point the midpoint between each successive pair of the sorted values. For each evaluation of a 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 candidate cut points. This 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.

FID is developed in the particular context of top-down induction of decision trees. It tends 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 (Quinlan, 1993). 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, hence is not subject to the same fragmentation problem if there are many values for an attribute. So FID's bias towards forming small number of intervals may not be so well justified for naive-Bayes classifiers as for decision trees (Yang & Webb, 2001).

Another unsuitability of FID for naive-Bayes classifiers is that FID discretizes a numeric attribute by calculating the class information entropy as if the naive-Bayes classifiers only use that *single* attribute after discretization. FID hence 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.

3.3 Lazy Discretization (LD)

LD (Hsu et al., 2000) delays probability estimation until classification time. It waits until a test instance is presented to determine the cut points for each numeric attribute. For a numeric attribute value 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 FKID 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 learning only keeps training instances to estimate the probabilities required by naive-Bayes classifiers during training time and discards them during classification time. In contrast, LD needs to keep training instances for use during classification time, which demands high memory, especially when the training data is large. Besides, LD defers until classification time estimating

[1] In practice, k is often set as 5 or 10.

$p(X_i = x_i \,|\, C = c)$ in (4) for each attribute of each test instance. In consequence, where a large number of instances need to be classified, LD will incur large computational overhead and execute the classification task much more slowly than eager approaches.

Although LD achieves comparable performance to FKID and FID (Hsu et al., 2000), the high memory and computational overhead might impede it from feasible implementation for classification tasks with large amounts of training or test data.

3.4 Proportional k-Interval Discretization (PKID)

PKID (Yang & Webb, 2001) adjusts discretization bias and variance by tuning the interval size and number, and further adjusts the probability estimation bias and variance of naive-Bayes classifiers to achieve lower classification error.

The larger the interval $(a_i, b_i]$, the more instances in it, the lower the discretization variance, hence the lower the variance of naive-Bayes' probability estimation. However, the larger the interval, the less distinguishing information is obtained about the particular value x_i of attribute X_i, the higher the discretization bias, hence the higher the bias of the probability estimation. So, increasing interval size (decreasing interval number) will decrease variance but increase bias. Conversely, decreasing interval size (increasing interval number) will decrease bias but increase variance.

PKID aims to resolve this conflict by adjusting the interval size and number proportional to the number of training instances. With the number of training instances increasing, both discretization bias and variance tend to decrease. Bias can decrease because the interval number increases. Variance can decrease because the interval size increases. This means PKID has greater capacity to take advantage of the additional information inherent in large volume of training data than either FKID or FID. FKID is fixed in the number of intervals, while FID tends to minimize the number of resulting intervals and does not tend to increase interval number accordingly.

Given a numeric attribute, supposing there are N training instances with known values for the attribute, the expected interval size is s (the number of instances in each interval) and the expected interval number is t, PKID employs (5) to calculate s and t:

$$\begin{aligned} s \times t &= N \\ s &= t. \end{aligned} \tag{5}$$

PKID then divides the sorted values into intervals, each of which containing s instances. Thus PKID seeks to give equal weight to discretization bias reduction and variance reduction by setting the interval size equal to the interval number ($s = t \approx \lfloor \sqrt{N} \rfloor$). Experiments show that PKID can significantly reduce classification error of naive-Bayes classifiers in comparison with FKID and FID.

4. Non-Disjoint Discretization

4.1 Overlapping Attribute Values

For a numeric attribute value x_i, discretization forms an interval $(a, b] \ni x_i$, and estimates $p(X_i = x_i | C = c)$ in (4) by

$$p(X_i = x_i | C = c) \approx p(a < X_i \leq b \,|\, C = c). \tag{6}$$

FKID, PKID and FID are *disjoint* discretizations, resulting in several *disjoint* intervals of a numeric attribute's values. No intervals overlap. When x_i is near either boundary of $(a, b]$, $p(a < X_i \leq b \,|\, C = c)$ is less likely to provide relevant information about x_i than when x_i is in the middle of $(a, b]$. In this situation, it is questionable to substitute $(a, b]$ for x_i.

In contrast, if discretization always locates x_i in the middle of $(a, b]$, it is reasonable to expect more distinguishing information about x_i by substituting $(a, b]$ for x_i, thus better classification performance from the resulting naive-Bayes classifiers. Lazy discretization (Hsu et al., 2000) embodies this idea to some extent. But we suggest that "lazy" learning is not necessary and propose the idea of Non-Disjoint Discretization (NDD), which "eagerly" discretizes a numeric attribute's values into overlapping intervals. Since the test instances are independent of each other, discretization does not need to form a uniform set of disjoint intervals for a numeric attribute, applying it to the attribute's entire range of values presented by the whole test instance set. Instead, it should form an interval most appropriate to the single value offered by the current test instance. Accordingly, for a numeric attribute, the discretized intervals formed for its two different values might *overlap* with each other because they are used for different test instances independently.

NDD can be implemented on the basis of FKID or PKID. Since previous experiments suggest that PKID reduces naive-Bayes classification error better than FKID does, we base the interval size determination strategy of NDD on that of PKID.

When discretizing a numeric attribute, given N, s, and t defined or calculated as in (5), NDD identifies among the sorted values t' *atomic intervals*,

$(a'_1, b'_1], (a'_2, b'_2], ..., (a'_{t'}, b'_{t'}]$, each with size equal to s', so that[2]

$$\begin{aligned} s' &= \frac{s}{3} \\ s' \times t' &= N. \end{aligned} \tag{7}$$

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 $(a'_{t'-2}, b'_{t'}]$. Figure 1 illustrates the procedure.

As a result, except in the case of falling into the first or the last *atomic interval*, a numeric value is always toward the middle of its corresponding interval. Thus, we prevent any numeric value from being near either boundary of its corresponding interval, making the probability estimation in (6) more reliable. Each interval has size equal to that of PKID by comprising three atomic intervals. Thus, we retain the advantage of PKID: giving the same weight to bias and variance reduction of naive-Bayes' probability estimation.

When implementing NDD, we follow rules listed below:

- Discretization is limited to known values of a numeric attribute. When applying (4) for an instance, we drop any attributes with an unknown value for this instance from the right-hand side.
- For some attributes, different training instances may hold identical values. We keep identical values in a single *atomic interval*. Thus although ideally each *atomic interval* should include exactly s' instances, its actual size may vary.
- We hold s' as the standard size of an *atomic interval*. We do not allow smaller size. We allow larger size only when it is because of the presence of identical values, or to accommodate the last interval if its size is between s' and $s' \times 2$.

4.2 Overlapping Attributes

There are interesting distinctions between our work and the work of **?** (**?**). For a numeric attribute X_i, **?** first find out its k cut points using either FID or FKID. Then X_i is substituted by k binary attributes so that the jth attribute corresponds to the test $X_i \leq cutpoint_j, (j = 1, \ldots, k)$. **?** thus convert each value into a bag of tokens. The closer two values are, the more overlapping their bags. The goal is to keep some ordinal information of each numeric attribute. The degree of 'overlap' measures the ordinal dissimilarity between two numeric values. This method is not designed for naive-Bayes classifiers. It creates a large number of inter-dependent attributes and hence is likely to compound naive-Bayes' attribute inter-dependence problem. In contrast, our approach involves only one attribute, with overlapping values, and hence does not add to the attribute inter-dependence problem.

*Figure 1. Atomic Interval*s Compose Actual Intervals

It is also interesting to contrast our work to the work of **?** (**?**). **?** propose to decompose an image into overlapping sub-regions when detecting faces or cars in the image. Each sub-region is an attribute and naive-Bayes classifiers are used to do classification. Like **?** (**?**), **?** use overlapping attributes while we use overlapping attribute values. **?**'s approach exacerbates the attribute inter-dependence problem while ours does not.

5. Algorithm Complexity Comparison

Suppose the number of training instances[3] and classes are n and m respectively. The complexity of each algorithm is as follows.

- NDD, PKID, and FKID are dominated by sorting. Their complexities are all $O(n \log n)$.
- FID 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 $O(mn \log n)$,

[2] Theoretically any odd number k is acceptable in 7 as long as the same number k of atomic intervals are grouped together later for the probability estimation. For simplicity, we take $k = 3$ for demonstration.

[3] Only instances with known values of the numeric attribute are under consideration.

which dominates the complexity of the sorting, resulting in complexity of order $O(mn \log n)$.

- LD performs discretization separately for each test instance and hence its complexity is $O(nl)$, where l is the number of test instances.

Accordingly NDD has the same order of complexity as PKID and FKID, and lower than FID and LD.

6. Experiments

We want to evaluate whether NDD can better reduce the classification errors of naive-Bayes classifiers, compared with FKID10 (FKID with $k = 10$), FID, PKID and LD. Suppose N is the number of training instances with known values of the numeric attribute to be discretized, the initial LD uses the same interval size as FKID10 (Hsu et al., 2000). Since according to previous experiments comparing PKID and FKID10, discretization with interval size $\lfloor \sqrt{N} \rfloor$ performs better than with interval size $\lfloor N/10 \rfloor$ (Yang & Webb, 2001), and NDD also adopts $\lfloor \sqrt{N} \rfloor$ as its interval size, for the sake of fair comparison, we include another version of lazy discretization, which uses $\lfloor \sqrt{N} \rfloor$ as the standard interval size. We name the initial lazy discretization LD10, while our new variation is LD$\sqrt{N}$.

6.1 Experimental Design

We run our experiments on 29 datasets from the UCI machine learning repository (Blake & Merz, 1998) and KDD archive (Bay, 1999), listed in Table 1. This experimental suite comprises 3 parts. The first part is composed of all the UCI datasets used by Fayyad and Irani (1993) when publishing the entropy minimization heuristic discretization. The second part is composed of all the datasets with numeric attributes used by Domingos and Pazzani (1997) for studying naive-Bayes classification. The third part is composed of larger datasets employed by Yang and Webb (2001) for evaluating the performance of PKID on larger datasets, so that we can check whether NDD maintains PKID's superiority on larger datasets by basing its interval size determination strategy on that of PKID's.

Table 1 gives a summary of the characteristics of each datasets, including the number of instances (Size), numeric attributes (Num.), categorical attributes (Cat.) and classes (Class). For each dataset, we implemented naive-Bayes classification by conducting a 10-trial, 3-fold cross validation. In each fold, we discretized the training data using each discretization method and applied the intervals so formed to the test data. We evaluated the performance of each method in terms of average classification error (the percentage of incorrect predictions) in the test across trials, which is also listed in Table 1, except that results could not be obtained for LD10 and LD$\sqrt{N}$ on the Forest-Covertype data due to excessive computation time.

6.2 Experimental Statistics

We employ three statistics to evaluate the experimental results in Table 1.

Mean error is the mean of errors across all datasets, except Forest-Covertype for which not all algorithms could complete. 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. Results are presented in the "ME" row of Table 1.

Geometric mean error ratio has been explained in detail by Webb (2000). It allows for the relative difficulty of error reduction in different datasets and can be more reliable than the mean error ratio across datasets. The "GM" row of Table 1 lists the geometric mean error ratios of PKID, FID, FKID10, LD10 and LD$\sqrt{N}$ against NDD respectively. Results for Forest-Covertype are excluded from this calculation because it could be completed neither by LD10 nor by LD$\sqrt{N}$.

Win/Lose/Tie record shows respectively the number of datasets for which NDD obtained better, worse or equal performance outcomes, compared with the alternative algorithms on a given measure. A one-tailed sign test can be applied to the record. If the sign test result is significantly low (here we use the 0.05 critical level), it is reasonable to conclude that it is unlikely that the outcome is obtained by chance and hence that the record of wins to losses represents a systematic underlying advantage to NDD with respect to the type of datasets on which they have been tested. The results are listed in Table 2.

6.3 Experimental Results

- NDD and LD$\sqrt{N}$ both achieve the lowest mean error among the six discretization methods.

- The geometric mean error ratios of the alternatives against NDD are all larger than 1. This suggests that NDD enjoys an advantage in terms of error reduction over the type of datasets studied in this research.

- With respect to the win/lose/tie records, NDD is better at reducing classification error than PKID,

Table 1. Experimental Datasets and Results

Dataset	Size	Num.	Cat.	Class	Error					
					NDD	PKID	FID	FKID10	LD10	LD$\sqrt{N}$
Labor-Negotiations	57	8	8	2	7.0	7.2	9.5	8.9	10.0	7.7
Echocardiogram	74	5	1	2	26.6	25.3	23.8	29.2	29.2	26.2
Iris	150	4	0	3	7.2	7.5	6.8	7.5	6.7	6.7
Hepatitis	155	6	13	2	14.4	14.6	14.5	14.7	14.2	14.2
Wine-Recognition	178	13	0	3	3.3	2.2	2.6	2.1	2.9	3.8
Sonar	208	60	0	2	26.9	25.7	26.3	25.2	26.4	27.3
Glass-Identification	214	9	0	3	38.8	40.4	36.8	39.4	22.0	22.2
Heart-Disease-(Cleveland)	270	7	6	2	18.6	17.5	17.5	17.1	17.6	17.8
Liver-Disorders	345	6	0	2	37.7	38.0	37.4	37.1	36.9	38.0
Ionosphere	351	34	0	2	10.2	10.6	11.1	10.2	10.8	11.3
Horse-Colic	368	7	14	2	20.0	20.9	20.7	20.9	20.8	19.7
Credit-Screening-(Australia)	690	6	9	2	14.4	14.2	14.5	14.5	13.9	14.7
Breast-Cancer-(Wisconsin)	699	9	0	2	2.6	2.7	2.7	2.6	2.6	2.7
Pima-Indians-Diabetes	768	8	0	2	25.8	26.3	26.0	25.9	25.4	26.4
Vehicle	846	18	0	4	38.5	38.2	38.9	40.5	38.1	38.7
Annealing	898	6	32	6	1.8	2.2	1.9	2.3	2.1	1.6
German	1000	7	13	2	25.4	25.5	25.1	25.4	25.3	25.0
Multiple-Features	2000	3	3	10	31.6	31.5	32.6	31.9	31.2	31.2
Hypothyroid	3163	7	18	2	1.7	1.8	1.7	2.8	2.3	1.7
Satimage	6435	36	0	6	17.5	17.8	18.1	18.9	18.4	17.5
Musk	6598	166	0	2	7.7	8.3	9.4	19.2	15.4	7.8
Pioneer-1-Mobile-Robot	9150	29	7	57	1.6	1.7	14.8	10.8	11.4	1.7
Handwritten-Digits	10992	16	0	10	12.1	12.0	13.5	13.2	12.8	12.1
Australian-Sign-Language	12546	8	0	3	35.8	35.8	36.5	38.2	36.4	35.8
Letter-Recognition	20000	16	0	26	25.6	25.8	30.4	30.7	27.9	25.5
Adult	48842	6	8	2	17.0	17.1	17.2	19.2	18.1	17.1
Ipums.la.99	88443	20	40	13	18.6	19.9	20.1	20.5	19.8	19.1
Census-Income	299285	8	33	2	23.3	23.3	23.6	24.5	25.0	23.6
Forest-Covertype	581012	10	44	7	31.4	31.7	32.1	32.9	-	-
ME	-	-	-	-	18.3	18.4	19.1	19.8	19.3	18.3
GM	-	-	-	-	1.00	1.01	1.11	1.16	1.14	1.01

FID, and FKID10 with frequency significant at 0.05 level.

- NDD is expected to maintain PKID's superior classification performance since it produces the same interval size as PKID does. It is further expected to exceed PKID since it produces more reliable probability estimation by setting values towards the middle of their intervals. This has been verified by the experiments. Compared with FID, among the 17 datasets where PKID wins, NDD also wins with only two exceptions. Among the 12 datasets where PKID does not win, NDD still can win 5 of them.

- NDD has more, but not significantly more wins than LD10 or LD$\sqrt{N}$ since they are similar in terms of locating an attribute value toward the middle of a discretized interval. But from the view of feasibility, NDD is overwhelmingly superior over LD algorithms. Table 3 lists out the computation time of training and testing a naive-Bayes classifier on data preprocessed by NDD, LD10 and LD$\sqrt{N}$ respectively in one fold out of 10-trial 3-fold cross validation for some larger datasets[4]. NDD is much faster than the LD algorithms.

[4]For Forest-Covertype, after 864000 seconds, neither LD10 nor LD$\sqrt{N}$ was able to obtain the classification results. Allowing for their feasibility for real-world classification tasks, it was meaningless to keep them running. So we stopped their processes, resulting in no precise records for the running time.

Table 2. Win/Lose/Tie Records

-	PKID	FID	FKID10	LD10	LD$\sqrt{n}$
NDD Win	19	20	22	15	14
NDD Lose	8	8	4	12	10
NDD Tie	2	1	3	1	4
Sign Test	0.03	0.02	0.01	0.35	0.27

Table 3. Running Time for One Fold (Seconds)

-	NDD	LD10	LD$\sqrt{N}$
Adult	0.7	6069	6025
Ipums.la.99	13	710607	691089
Census-Income	10	415026	510859
Forest-Covertype	60	> 864000	> 864000

- LD$\sqrt{N}$ has lower error than LD10 more often than the reverse, with Win/Lose/Tie record 15/10/3, although a sign test reveals that this is not significant at 0.05 level (Sign Test = 0.21).

7. Conclusion

Unlike many other learning algorithms, naive-Bayes classifiers do not require that the values of a numeric attribute be disjoint. Rather, due to the independence assumption, they require that only one value of an attribute be utilized for classification of a given instance. Based on this insight, we have proposed a new discretization method, Non-Disjoint Discretization (NDD). NDD forms a series of overlapping intervals for a numeric attribute. When discretizing a value of a numeric attribute for a given instance, it selects the discretized interval that places this value toward its center. This makes the substitution of the interval for the value more reliable for Naive-Bayes' probability estimation. It also seeks a good trade-off between the bias and variance of naive-Bayes' probability estimation by adjusting both the number and size of the discretized intervals to the quantity of the training data provided. Our experiments with an extensive selection of UCI and KDD datasets suggest that NDD provides lower classification error for Naive-Bayes classifiers than PKID, FID and FKID. It delivers comparable classification accuracy to the lazy discretization techniques but with much greater efficiency.

References

Bay, S. D. (1999). The UCI KDD archive [http://kdd.ics.uci.edu]. Department of Information and Computer Science, University of California, Irvine.

Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/mlrepository.html]. Department of Information and Computer Science, University of California, Irvine.

Catlett, J. (1991). On changing continuous attributes into ordered discrete attributes. *Proceedings of the European Working Session on Learning* (pp. 164–178).

Cestnik, B. (1990). Estimating probabilities: A crucial task in machine learning. *Proceedings of the European Conference on Artificial Intelligence* (pp. 147–149).

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., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *Proceedings of the Twelfth International Conference on Machine Learning (ICML'95)* (pp. 194–202). Morgan Kaufmann Publishers, Inc.

Fayyad, U. M., & Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. *IJCAI-93: Proceedings of the 13th International Joint Conference on Artificial Intelligence* (pp. 1022–1027). Morgan Kaufmann Publishers, Inc.

Hsu, C.-N., Huang, H.-J., & Wong, T.-T. (2000). Why discretization works for naive Bayesian classifiers. *Machine Learning, Proceedings of the Seventeenth International Conference (ICML'2000)* (pp. 309–406). Morgan Kaufmann Publishers, Inc.

John, G. H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers, Inc.

Johnson, R., & Bhattacharyya, G. (1985). *Statistics: Principles and methods*. John Wiley & Sons.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers, Inc.

Scheaffer, R. L., & McClave, J. T. (1995). *Probability and statistics for engineers*, chapter Continuous Probability Distributions, 186. Duxbury Press. Fourth edition.

Webb, G. I. (2000). Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, *40*, 159–196.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR1999)* (pp. 42–49).

Yang, Y., & Webb, G. I. (2001). Proportional k-interval discretization for naive-Bayes classifiers. *12th European Conference on Machine Learning (ECML'01)* (pp. 564–575). Springer.