Pre-publication draft of the paper which appeared in the Proceedings off the 12th European Conference on Machine Learning (ECML01) pp 564-575

Proportional k-Interval Discretization for Naive-Bayes Classifiers

Ying Yang and Geoffrey I. Webb

School of Computing and Mathematics, Deakin University, Vic3217, Australia

Abstract. This paper argues that two commonly-used discretization approaches, fixed k-interval discretization and entropy-based discretization have sub-optimal characteristics for naive-Bayes classification. This analysis leads to a new discretization method, Proportional k-Interval Discretization (PKID), which adjusts the number and size of discretized intervals to the number of training instances, thus seeks an appropriate trade-off between the bias and variance of the probability estimation for naive-Bayes classifiers. We justify PKID in theory, as well as test it on a wide cross-section of datasets. Our experimental results suggest that in comparison to its alternatives, PKID provides naive-Bayes classifiers competitive classification performance for smaller datasets and better classification performance for larger datasets.

1 Introduction

Many real-world classification tasks involve numeric attributes. Consequently, appropriate handling of numeric attributes is an important issue in machine learning. For naive-Bayes classifiers, numeric attributes are often processed by discretization. For each numeric attribute A, a new nominal attribute A^* is created. Each value of A^* corresponds to an interval of the numeric values of A. When training a classifier, the learning process uses the nominal A^* instead of the original numeric A.

A number of discretization methods have been developed. One common approach is fixed k-interval discretization [1, 2, 3, 4, 5]. It directly discretizes values of a numeric attribute into k equal-width intervals. Another approach uses information measures to discretize a numeric attribute into intervals. For example, Fayyad & Irani's entropy minimization heuristic [6] is extensively employed. Each of these strategies has advantages and disadvantages. Fixed k-interval discretization is easy to implement. But it does not adjust its behavior to the specific characteristics of the training data. Fayyad & Irani's heuristic approach was developed in the context of decision tree learning. It seeks to identify a small number of intervals, each dominated by a single class. However, for naive-Bayes classification, in contrast to decision tree learning, it is plausible that it is less important to minimize the number of intervals or to form intervals dominated by a single class.

In this paper, we introduce a new approach for discretizing numeric attributes. We focus our attention on classification tasks using naive-Bayes classifiers. We seek to balance two conflicting objectives. On one hand, we prefer

forming as many intervals as possible. This increases the representation power of the new nominal attribute. That is, the more intervals formed, the more distinct values the classifier can distinguish between. On the other hand, we should ensure that there are enough training instances in each interval, so that we have enough information to accurately estimate the probabilities required by Bayes' theorem. But when we are training a classifier, we usually have a fixed number of training instances. The number of intervals will decrease when the size of intervals (the number of instances in each interval) increases and vice versa. This can be viewed as a bias-variance [7] trade-off. Increasing the number of intervals will decrease bias and increase variance and vice versa. Allowing for this, we propose Proportional k-Interval Discretization (PKID). This strategy adjusts the number and size of discretized intervals proportional to the number of training instances, seeking an appropriate trade-off between the granularity of the intervals and the expected accuracy of probability estimation. Currently we adopt a compromise: given a numeric attribute A for which the number of instances that have a known value is N, we take the proportional coefficient as $\sqrt{N}$. We discretize A into $\sqrt{N}$ intervals, with $\sqrt{N}$ instances in each interval. Thus, both objectives receive the same weight. As N increases, both the number and size of discretized intervals increase. These are very desirable characteristics that we will discuss in more detail later.

To evaluate this new technique, we separately implement PKID, Fayyad & Irani's discretization (FID), and fixed k-interval discretization (FKID) with k=5,10 to train naive-Bayes classifiers. We compare the classification errors of the resulting classifiers. Our hypothesis is that naive-Bayes classifiers trained on data formed by PKID will have competitive classification error to those trained on data formed by alternative discretization approaches for smaller datasets, and that PKID will be able to utilize the incremental information in larger datasets to achieve lower classification error.

The rest of this paper is organized as follows. We give an overview of naive-Bayes classifiers and discretization in Section 2 and 3 respectively. In Section 4, we discuss Proportional k-Interval Discretization in detail. We compare the algorithm complexities in Section 5. Experimental results are presented in Section 6. Section 7 provides a conclusion and suggests research directions that are worth further exploration.

2 Naive-Bayes Classifiers

Naive-Bayes classifiers are simple, efficient and robust to noise and irrelevant attributes. One defect, however, is that naive-Bayes classifiers utilize an assumption that the attributes are conditionally independent of each other given the class. Although this assumption is often violated in the real world, the classification performance of naive-Bayes classifiers is still surprisingly good for many classification tasks, compared with other more complex classifiers. According to [8], this is explained by the fact that classification estimation is only a function of

the sign (in binary cases) of the function estimation; the classification accuracy can remain high even while function approximation is poor.

We briefly introduce the main idea of naive-Bayes classifiers as follows. 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 training instances with their class labels, the training dataset, is provided, and a new instance is presented. The learner is asked to predict the class for this new instance according to the evidence provided by the training dataset. We define:

- C as the 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 $X_1 = x_1 \wedge X_2 = x_2 \wedge \cdots \wedge X_k = x_k$.

Bayes' theorem can be used to calculate the probability of each class given the instance x:

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

Expected error can be minimized by choosing the class with the highest probability as the class of the instance x. Because the probabilities needed by the calculation are not known, it is necessary to estimate them from the training dataset. Unfortunately, since x is usually an unseen instance which does not appear in the training dataset, it may not be possible to directly estimate $p\,(X = x \,|\, C = c)$. So a simplification is made: if each attribute $X_1, X_2, \cdots, X_k$ is conditionally independent of each other given the class, then:

$$\begin{aligned} p\,(X = x \,|\, C = c) &= p\,(\wedge X_i = x_i \,|\, C = c) \\ &= \prod p\,(X_i = x_i \,|\, C = c) \,. \end{aligned} \tag{2}$$

Since the denominator in formula 1, $p\,(X = x)$, is invariant across classes, it does not affect the final choice and can be dropped. Thus one can further estimate the most probable class using:

$$p\,(C = c \,|\, X = x) \propto p\,(C = c) \prod p\,(X_i = x_i \,|\, C = c) \,. \tag{3}$$

Classifiers using the independence assumption embodied in formula 2 are called naive-Bayes classifiers. The independence assumption 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 [9].

3 Discretize Numeric Attributes

An attribute is either nominal or numeric. Values of a nominal attribute are discrete. Values of a numeric attribute are either discrete or continuous [10].

For each attribute X_i with value x_i, $p(X_i = x_i \mid C = c)$ in formula 2 is often 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 valued 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 formula 3. In keeping with normal terminology for this research area, we call the conversion of a numeric attribute to a nominal attribute, *discretization*, irrespective of whether this numeric attribute is discrete or continuous.

A nominal attribute usually takes only a small number of values. The probabilities $p(X_i = x_i \mid C = c)$ and $p(C = c)$ can be estimated from the frequencies of $X_i = x_i \wedge C = c$ and $C = c$ in the training dataset. In our experiment, when $p(X_i = x_i \mid C = c)$ was estimated, the M-estimate [11] with m=2 was used. When $p(C = c)$ was estimated, the Laplace-estimate [11] was used.

- M-estimate: $\frac{n_{ci}+mp}{n_c+m}$, where n_{ci} is the number of instances that satisfy $X_i = x_i \wedge C = c$, n_c is the number of instances that satisfy $C = c$, p is the prior estimate of $X_i = x_i$, $p(X_i = x_i)$ (estimated by Laplace-estimate), and m is a constant (2 in our research).
- Laplace-estimate: $\frac{n_c+k}{N+n*k}$, where n_c is the number of instances that satisfy $C = c$, n is the number of classes, N is the number of training instances, and k is normally 1.

A continuous numeric attribute has an infinite number of values, as do many discrete numeric attributes. The values are generated according to some probability distribution. Since classification tasks are normally carried out for real-world data, whose real probability distribution is unknown, a difficulty in naive-Bayes classification is how to estimate $p(X_i = x_i \mid C = c)$ when X_i is numeric. A common solution is discretization [12]. Discretization transforms numeric attributes into nominal attributes before they are used to train classifiers. In consequence, they are not bound by some specific distribution assumption. But since we do not know the real relationship underlying different values of a numeric attribute, discretization may suffer from loss of information.

One common discretization approach is fixed k-interval discretization (FKID). It divides a numeric attribute into k intervals, where (given n observed instances) each interval contains n/k (possibly duplicated) adjacent values. Here k is determined without reference to the properties of the training data[1]. A problem of

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

this method is that it ignores relationships among different values, thus potentially suffering much attribute information loss. But although it may be deemed inelegant, this simple discretization technique works surprisingly well for naive-Bayes classifiers. Hsu, Huang and Wong [13] provided an interesting analysis of the reason why fixed k-interval discretization works for naive-Bayes classifiers. They suggested that discretization approaches usually assume that discretized attributes have Dirichlet priors. "Perfect Aggregation" of Dirichlets can ensure that naive-Bayes with discretization appropriately approximates the distribution of a numeric attribute.

Another popular discretization approach is Fayyad & Irani's entropy minimization heuristic discretization (FID) [6]. They first suggested binary discretization, which discretizes values of a numeric attribute into two intervals. The training instances are first sorted by increasing values of the numeric attribute, and the midpoint between each successive pair of attribute values in the sorted sequence is evaluated as a potential cut point. FID selects the "best" cut point from the range of values by evaluating every candidate cut point. For each evaluation of a candidate cut point, the data are discretized into two intervals and the class information entropy of the resulting discretization is computed. A binary discretization is determined by selecting the cut point for which the entropy is minimal amongst all candidate cut points. Later, they generalized the algorithm to multi-interval discretization. The training instances are sorted once, then the binary discretization is applied recursively, always selecting the best cut point. A minimum description length criterion is applied to decide when to refrain from applying further binary discretization to a given interval.

FID was presented in the particular context of top-down induction of decision trees. It tends to form nominal 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 [14]. 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 considers attributes independent of one another given the class, hence is not subject to the same fragmentation problem as experienced in decision tree learning if there are many values for a single attribute. So aiming at minimizing the number of discretized intervals for naive-Bayes classifiers may not be as well justified as for decision trees.

4 Proportional k-Interval Discretization

The conditional probabilities in formula 3 of a numeric attribute x will be drawn from an unknown probability density function $f(x|y)$. If we form a discretized value x_i^* corresponding to the interval $(a, b]$ of x, then

$$p(x_i^*|y) = \int_a^b f(x|y)\, \mathrm{d}x. \tag{4}$$

We wish to estimate $p(x_i^*|y)$ from data. The larger the interval $(a, b]$, the more instances will be contained in it, and the lower the variance of the probability estimation. Conversely, however, the larger the interval, the less distinguishing information is obtained about each particular value of x, and hence the higher the bias of the probability estimation. So, on one hand we wish to increase the range of values in each interval in order to decrease variance, and on the other hand we wish to decrease the range of values to decrease bias.

We suggest Proportional k-Interval Discretization (PKID). This strategy seeks an appropriate trade-off between the bias and variance of the probability estimation by adjusting the number and size of intervals to the number of training instances. Currently we adopt a compromise: given a numeric attribute, supposing we have N training instances with known values for the attribute, we discretize it into $\sqrt{N}$ intervals, with $\sqrt{N}$ instances in each interval[2]. Thus we give equal weight to both bias and variance management. Further, with N increasing, both the number and size of intervals increase correspondingly, which means discretization can decrease both the bias and variance of the probability estimation. This is very desirable, because if a numeric attribute has more instances available, there is more information about it. A good discretization scheme should respond to this increase in information accordingly. But fixed k-interval discretization is fixed in the number of intervals and does not react to the above-mentioned information increase. Fayyad & Irani's discretization tends to minimize the number of resulting intervals, as is appropriate in order to avoid the fragmentation problem in decision tree learning, and does not tend to increase the number of intervals accordingly.

When implementing PKID, we follow rules listed below:

- Discretization is limited to known values of a numeric attribute. We ignore any unknown values. When applying formula 3 for a testing 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 always keep the identical values in a single interval. Thus although ideally each interval should include exactly $\sqrt{N}$ instances, the actual size of each interval may vary.
- Given N training instances with known values of a numeric attribute, we hold $\lfloor\sqrt{N}\rfloor$ as the standard size of the discretized interval (the number of instances in an interval should be an integer). 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 when its size is between $\lfloor\sqrt{N}\rfloor$ and $\lfloor\sqrt{N}\rfloor \times 2$.

[2] We do not form intervals based on the number of values, such as, creating intervals with m values each. As Catlett [2] pointed out, this type of discretization is vulnerable to outliers that may drastically skew the range.

5 Algorithm Complexity Comparison

Each of the discretization algorithms, PKID, FID, and FKID can be considered to be composed of two stages. The first stage is to sort a numeric attribute by increasing values. The second stage is to discretize the sorted values into intervals. Suppose the number of training instances with known values of the attribute is n, and the number of classes is m. The complexity of each algorithm is as follows.

- PKID and FKID are dominated by sorting values of an attribute. So their complexities are all $O(n \log n)$.
- FID also does sorting first, resulting in $O(n \log n)$. 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. Thus finding the cut points is an operation with maximum complexity $O(mn \log n)$. So FID's maximum complexity is $O(mn \log n)$.

This means that PKID has the same order of complexity as FKID, and lower than FID.

6 Experiments

We want to evaluate whether or not PKID can better reduce the classification error of naive-Bayes classifiers, compared with 5D, 10D (FKID with k=5, 10) and FID.

6.1 Experimental Design

We ran our experiments on 31 natural datasets from the UCI machine learning repository [15] and KDD archive [16], listed in Table 2. They exhibit a range of different sizes. This experimental suite comprises 3 parts. The first part is composed of all the UCI datasets used by [6] when publishing the entropy minimization heuristic for discretization. The second part is composed of all the UCI datasets with numeric attributes used by [17] for studying naive-Bayes classification. In addition, as PKID responds to dataset size, and the first two parts contain mainly datasets with relatively few instances, we further augmented this collection with datasets that we could identify containing numeric attributes, with emphasis on those having more than 5000 instances. The performance of PKID will differ most substantially from those of the alternatives when there are many instances in the training dataset and hence many intervals are formed. Therefore, if the technique is successful, we can expect PKID to demonstrate the greatest advantage for larger datasets.

Table 2 lists an index[3], as well as the number of instances (Size), numeric attributes (Num.), nominal attributes (Nom.) and classes (Class) for each dataset. For each dataset, a 10-trial, 3-fold cross validation is used to train and test a naive-Bayes classifier. In each fold, the dataset was discretized separately by the above-mentioned four approaches. Thus we obtained four versions of the original dataset. For each version, a naive-Bayes classifier was learned. We evaluated its classification performance in terms of average error (the percentage of incorrect classifications) in the testing data across trials. The testing data was not available to the discretization algorithm during discretization. Discretization was performed only by reference to the training data for a given cross validation fold.

The classification errors of PKID, FID, 10D and 5D on each dataset are also listed in Table 2. The records are sorted in ascending order of the datasets' sizes, so that we can track the effect of dataset size on PKID's performance. In each record, **boldface** font indicates the algorithm achieving the best classification performance for this dataset.

6.2 Experimental Statistics

We employed three statistics to evaluate the experimental results in Table 2.

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. The mean error for each algorithm is presented in the "Mean Error" row of Table 2.

Geometric mean error ratio. This method has been explained in detail by [18]. 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. The geometric mean error ratio of algorithm X against algorithm Y, $GM(X,Y)$, is calculated as

$$GM(X,Y) = \sqrt[n]{\prod_{i=1}^{n} \frac{x_i}{y_i}} ,$$

where x_i and y_i are respectively the errors of algorithm X and algorithm Y for the ith dataset, and n is the number of the employed datasets. The last row of Table 2 lists out the geometric mean error ratios of PKID against FID, 10D and 5D.

Win/Lose/Tie record. The three values are, respectively, the number of datasets for which PKID obtained better, worse or equal performance outcomes, compared with the alternative algorithms on a given measure. A sign test can be applied to these summaries. 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 one of the algorithms

[3] For reference from Fig. 1.

with respect to the type of datasets on which they have been tested. These win/lose/tie records and the sign test results are summarized in Table 1.

-	FID	10D	5D
PKID Win	21	21	23
PKID Lose	7	7	7
PKID Tie	3	3	1
Sign Test	≤ 0.0063	≤ 0.0063	≤ 0.0026

Table 1. Win/Lose/Tie

6.3 Experimental Evaluations

Utilizing the above statistics, we have the following evaluations:

- PKID achieves the lowest mean error among the four discretization approaches.
- The geometric mean error ratios of PKID against FID, 10D and 5D are all less than 1. This suggests that PKID 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, PKID is significantly better than all of FID, 10D and 5D in terms of reducing classification errors.
- PKID demonstrates advantage more apparently as datasets become larger. For datasets containing more than 1000 instances, it is only outperformed in Hypothyroid. For datasets containing fewer than 1000 instances, the win/lose/tie records of PKID against FID, 10D and 5D are respectively 10/6/2, 8/7/3, and 10/7/1, suggesting that PKID has at worst comparable performance to these alternatives. This tendency, which is also illustrated in Figure 1, results from the ability of PKID to take advantage of training information increase by adjusting the size and number of discretized intervals to the number of training instances, thus achieves better classification performance among larger datasets.
- We suggest that PKID can adjust the number of discretized intervals to the number of training instances. To show a gross profile, the last two columns of Table 2 list the mean number of intervals produced by PKID and FID for each dataset, averaged on all the numeric attributes across 10 trials $\times$ 3 folds. Apparently, PKID is more sensitive to the increase of training instances than FID.

7 Conclusions and Further Research

In this paper, we reviewed two common-used discretization approaches for naive-Bayes classifiers, FKID and FID. We then proposed a new discretization method,

Fig. 1. PKID Responds to Dataset Size

Proportional k-Interval Discretization (PKID). We argue PKID is more appropriate than FKID and FID for naive-Bayes classifiers. It attaches importance to both the number and size of discretized intervals, and adjusts them in response to the quantity of training data provided.

In our research, we have used $\sqrt{N}$ as the size of intervals to be formed. This was selected as a means to provide equal weight to both bias and variance managements. A promising direction for further research is to investigate alternative approaches to adjust interval size to training dataset size. It is plausible that selection of interval size should be responsive to some other attributes of the training dataset. For example, it might be that the more classes a dataset contains, the larger the optimal interval size, as more data is required for accurate conditional probability estimation. It may also be that as dataset size increases, there is greater potential for gains through one rather than the other of the two objectives, bias reduction and variance reduction, and hence the interval size should be weighted to favor one over the other.

Our experiments with an extensive selection of UCI and KDD datasets suggest that in comparison to its alternatives, PKID provides naive-Bayes classifiers competitive classification performance for smaller datasets and better classification performance for larger datasets.

References

[1] Wong, A. K. C., Chiu, D. K. Y.: Synthesizing Statistical Knowledge from Incomplete Mixedmode Data, IEEE Transaction on Pattern Analysis and Machine Intelligence 9, 796-805, 1987

[2] Catlett, Jason: Megainduction: Machine Learning on Very Large Databases, University of Sydney, Australia, 1991
[3] Catlett, Jason: On Changing Continuous Attributes into Ordered Discrete Attributes, Proceedings of the European Working Session on Learning, 164-178, 1991
[4] Chmielewski, M. R., Grzymala-Busse, J. W.: Global Discretization of Continuous Attributes as Preprocessing for Machine Learning, Third International Workshop on Rough Sets and Soft Computing, 294-301, 1994
[5] Pfahringer, Bernhard: Compression-Based Discretization of Continuous Attributes, Proceedings of the Twelfth International Conference on Machine Learning, 1995
[6] Fayyad, Usama M., Irani, Keki B.: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, Proceedings of the 13th International Joint Conference on Artificial Intelligence, 1022-1027, 1993
[7] Kohavi, R., Wolpert, D.: Bias Plus Variance Decomposition for Zero-One Loss Functions, Proceedings of the 13th International Conference on Machine Learning, 275-283, 1996
[8] Domingos, Pedro, Pazzani, Michael: On the Optimality of the Simple Bayesian Classifier under Zero-One Loss, Machine Learning 29, 103-130, 1997
[9] Yang, Yiming, Liu, Xin: A Re-examination of Text Categorization Methods, Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval, 42-49, 1999
[10] Johnson, Richard, Bhattacharyya, Gouri: Statistics: Principles and Methods, 12-13, 1985
[11] Cestnik, B.: Estimating Probabilities: A Crucial Task in Machine Learning, Proceedings of the European Conference on Artificial Intelligence, 147-149, 1990
[12] Dougherty, James, Kohavi, Ron, Sahami, Mehran: Supervised and Unsupervised Discretization of Continuous Features, Proceedings of the Twelfth International Conference on Machine Learning, 194-202, 1995
[13] Hsu, Chun-Nan, Huang, Hung-Ju, Wong, Tzu-Tsung: Why Discretization works for Naive Bayesian Classifiers, Machine Learning, Proceedings of the Seventeenth International Conference, 309-406, 2000
[14] Quinlan, J. Ross: C4.5: Programs for Machine Learning, 1993
[15] Blake, C. L., Merz, C. J.: UCI Repository of Machine Learning Databases [http://www.ics.uci.edu/∼mlearn/MLRepository.html], Department of Information and Computer Science, University of California, Irvine, 1998
[16] Bay, S. D.: The UCI KDD Archive [http://kdd.ics.uci.edu], Department of Information and Computer Science, University of California, Irvine, 1999
[17] Domingos, Pedro, Pazzani, Michael: Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier, Proceedings of the Thirteenth International Conference on Machine Learning, 105-112, 1996
[18] Webb, Geoffrey I.: MultiBoosting: A Technique for Combining Boosting and Wagging, Machine Learning, 40-2, 159-196, 2000

Index	Dataset	Size	Num.	Nom.	Class	Error (%)				Inter. No.	
						PKID	FID	10D	5D	PKID	FID
A	Labor Negotiations	57	8	8	2	7.7	9.5	9.6	**7.5**	2	2
B	Echocardiogram	74	5	1	2	26.5	**23.8**	29.2	25.4	5	2
C	Postoperative Patient	90	1	7	3	**36.1**	36.3	**36.1**	36.3	2	2
D	Iris	150	4	0	3	7.5	**6.8**	7.5	7.6	7	4
E	Hepatitis	155	6	13	2	14.4	14.5	14.7	**14.3**	7	2
F	Wine Recognition	178	13	0	3	**2.1**	2.6	**2.1**	2.2	9	4
G	Sonar	208	60	0	2	25.4	26.3	25.2	**24.0**	10	2
H	Glass Identification	214	9	0	3	**24.1**	24.9	24.8	27.9	8	3
I	Heart Disease (Cleveland)	270	7	6	2	17.5	17.5	**17.1**	17.2	8	2
J	Liver Disorders	345	6	0	2	38.0	37.4	37.1	**34.5**	10	2
K	Ionosphere	351	34	0	2	10.6	11.1	**10.1**	11.9	12	4
L	Horse Colic	368	8	13	2	20.9	**20.6**	20.8	20.9	7	2
M	Synthetic Control Chart	600	60	0	6	**2.4**	2.8	3.4	5.3	19	5
N	Credit Screening (Australia)	690	6	9	2	14.2	14.5	14.5	**14.1**	15	3
O	Breast Cancer (Wisconsin)	699	9	0	2	2.7	2.7	**2.6**	3.2	6	4
P	Pima Indians Diabetes	768	8	0	2	26.1	26.0	**25.9**	26.8	16	3
Q	Vehicle	846	18	0	4	**38.3**	38.9	40.5	43.6	16	5
R	Annealing	898	6	32	6	4.8	**2.8**	7.7	8.9	5	3
S	German	1000	7	13	2	**25.1**	**25.1**	25.4	25.2	9	2
T	Multiple Features	2000	3	3	10	**31.5**	32.6	31.9	33.4	35	6
U	Hypothyroid	3163	7	18	2	1.8	**1.7**	2.8	4.3	27	4
V	Satimage	6435	36	0	6	**17.8**	18.1	18.9	20.6	34	6
W	Musk	6598	166	0	2	**8.3**	9.4	19.2	25.7	47	5
X	Pioneer-1 Mobile Robot	9150	29	7	57	**1.7**	14.8	10.8	21.9	37	5
Y	Handwritten Digits	10992	16	0	10	**12.0**	13.5	13.2	15.9	47	5
Z	Australian Sign Language	12546	8	0	3	**35.8**	36.5	38.2	42.5	19	4
1	Letter Recognition	20000	16	0	26	**25.8**	30.4	30.7	38.2	11	5
2	Adult	48842	6	8	2	**17.1**	17.2	19.2	19.2	49	5
3	Ipums.la.99	88443	20	40	13	**19.9**	20.1	20.5	20.4	29	4
4	Census Income	299285	8	33	2	**23.3**	23.6	24.5	25.0	80	5
5	Forest Covertype	581012	10	44	7	**31.7**	32.1	32.9	32.6	264	6
-	**Mean Error**	-	-	-	-	18.4	19.2	19.9	21.2	-	-
-	**Geometric Mean Error Ratio**	-	-	-	-	1.00	0.92	0.85	0.78	-	-

Table 2. Experimental Datasets and Results