Naive Bayes Classifiers that Perform Well with Continuous Variables

Remco R. Bouckaert

Computer Science Department, University of Waikato &
Xtal Mountain Information Technology
New Zealand
remco@cs.waikato.ac.nz, rrb@xm.co.nz**

Abstract. There are three main methods for handling continuous variables in naive Bayes classifiers, namely, the normal method (parametric approach), the kernel method (non parametric approach) and discretization. In this article, we perform a methodologically sound comparison of the three methods, which shows large mutual differences of each of the methods and no single method being universally better. This suggests that a method for selecting one of the three approaches to continuous variables could improve overall performance of the naive Bayes classifier. We present three methods and discuss efficient ways to implement v-fold cross validation for the normal, kernel and discretization method. Empirical evidence suggests that selection using 10 fold cross validation (especially when repeated 10 times) can largely and significantly improve over all performance of naive Bayes classifiers and consistently outperform any of the three popular methods for dealing with continuous variables on their own. This is remarkable, since selection among more classifiers does not consistently result in better accuracy.

1 Introduction

Naive Bayes classifiers perform well over a wide range of classification problems, including medical diagnosis, text categorization, collaborative and email filtering, and information retrieval (see [14] for a pointer to the literature). Compared with more sophisticated schemes, naive Bayes classifiers often perform better [5]. Furthermore, naive Bayes can deal with a large number of variables and large data sets, and it handles both discrete and continuous attribute variables.

There are three main methods for dealing with continuous variables in naive Bayes classifiers (Section 2 has the technical details). The *normal method* is the classical method that approximates the distribution of the continuous variable using a parameterized distribution such as the Gaussian. The *kernel method* [9] uses a non-parameterized approximation. Finally, the *discretization methods* [6] first discretizes the continuous variables into discrete ones, leaving a simpler

** Please cite as; Remco R. Bouckaert. Naive Bayes Classifiers that Perform Well with Continuous Variables. In Proceedings of the 17th Australian Conference on AI (AI 04), Lecture Notes AI, Berlin: Springer. 2004.

problem without any continuous variables. In recent years, much progress has been made in understanding why discretization methods work [8, 14, 15]. In general, it is acknowledged that the normal method tends to perform worse than the other two methods.

However, experimental comparisons among all three methods have not been performed. Also, comparisons among the normal with kernel methods [9] and normal with discretization methods [6] is based on single runs of cross validation, which is known to have an elevated Type I error [4] and suffers from low replicability [3]. In this article, we perform a proper comparison and show how to select the method that performs well using an efficient method.

Unfortunately, previous work on naive Bayes classifiers dealing with continuous variables comparing the three methods mentioned is supported by experiments with flawed designs. John and Langely [9] made experimental claims about the benefits of kernel based naive Bayes based on 10 fold cross validation. This method has an elevated Type I error [4] and furthermore has very low replicability. In fact, for problems where learning algorithms are compared that differ only slightly in performance, the probability that an experiment gives the same outcome twice when repeated with the same data, the same algorithms, the same hypothesis test with the same parameters, but only a different randomization of the data is as low as two out of three [3]. Dougherty et al. [6] claimed that discretization using MDL [7] outperforms the normal method. However, their experimental design is based on 5 fold cross validation, which suffers from an elevated Type I error and low replicability as well.

In the following section, we describe the classification problem, give a technical description of naive Bayes classifiers and the three methods for dealing with continuous variables. Section 3 deals with selecting the correct method for a given data set and shows how to do this efficiently. In Section 4, we present experimental results and discuss our findings. We conclude with recommendations and directions for further research in Section 5.

2 Naive Bayes classifiers

Let $\mathbf{U}$[1] be a set of variables. One variable $Y \in \mathbf{U}$ of particular interest is called the *class variable* with class values $\{y_1, \ldots, y_k\}$ $(k > 1)$. The remaining variables, $\mathbf{U}\backslash\{y\} = \mathbf{X} = \{X_1, \ldots, X_n\}$ $(n > 0)$ are called *attribute variables*, which can be discrete or continuous. The classification problem consists of finding a mapping $\hat{y} = f(\mathbf{X})$ from the attributes to the class variable based on data sets D over $\mathbf{U}$ such that for an unseen future instance of $\mathbf{X}$ we can predict the value of Y with high expected accuracy $E\{\hat{y} = y\}$ (assuming D is a representative sample for the problem).

The naive Bayes classifier solves the problem by defining a probability distribution $P(\mathbf{U})$ over $\mathbf{U}$. Learning the complete distribution is in general impracti-

[1] We use bold upper case to denote sets of variables, upper case to denote single variables, lower case to denote variable values and lower case bold to denote values of sets of variables.

cal, so the assumption is made that all attributes are conditionally independent given the class variable. This allows us to write the distribution $P(\mathbf{U})$ as the product $P(Y)\prod_{X\in\mathbf{X}} P(X|Y)$. The naive Bayes method classifies $\mathbf{x}$ by taking the value of Y with the highest probability given $\mathbf{X}$. So $\hat{y} = \text{argmax}_{y\in\{y_1,\ldots,y_k\}} P(Y = y|\mathbf{X} = \mathbf{x}) \propto \text{argmax}_{y\in\{y_1,\ldots,y_k\}} P(Y = y)\prod_{X\in\mathbf{X}} P(X = x|Y = y)$.

Learning a naive Bayes classifier from a data set D breaks down to estimating the distributions $P(Y)$ and $P(X|Y)$ for each $X \in \mathbf{X}$. The distribution over the class can be directly estimated from D as $P(Y = y) = 1+\#$ (cases in D with $Y = y)/k+\#$ (cases in D). The 1 in numerator and k in denominator are a Laplace correction to ensure no zero-counts are encountered[2]. When X is discrete with k_x values, we can likewise estimate $P(X = x|Y = y)$ simply by

$$P(X = x|Y = y) = \frac{1 + \#\text{ (cases in } D \text{ with } Y = y \text{ and } X = x)}{k_x + \#\text{ (cases in } D \text{ with } Y = y)} \tag{1}$$

When X is a continuous variable, there are essentially three methods for estimating $P(Y|X)$, namely, using a parameterized distribution, a non-parameterized distribution and by discretization after which it can be dealt with as a discrete variable.

2.1 Normal method

The classic method to approximate $P(X|Y)$ for a continuous variable X is assuming that it follows a known distribution for which parameters can be estimated from the data. The most popular assumption is that the distribution follows the Gaussian aka normal distribution, hence we call this the *normal method*. For each of the class values y, we assume that $P(X = x|Y = y) = N(x|m_y, \sigma_y)$ where $N(x|m, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-m)^2}{2\sigma^2}}$ is the normal distribution with mean m and variance σ.

Let x_i $(0 < i \leq n)$ be the value of X in the ith record of D and likewise y_i the value of Y. The means m_y and variances σ_y are estimated from the values of X in instances for which $Y = y$ simply through unbiased estimates $m_y = \frac{1}{\#y}\sum_{i=1,y_i=y}^{n} x_i$ and $\sigma_y^2 = \frac{1}{\#y-1}\sum_{i=1,y_i=y}^{n}(x_i^2 - m_y^2)$ where $\#y$ is the number of instances in D where Y has value y.

The benefit of the normal method is that it performs well if the underlying distribution indeed follows a normal distribution [9], is fast in both learning and classification time and requires little memory.

2.2 Kernel method

The *kernel method* approximates $P(X|Y)$ for a continuous variable X by a sum of so called kernels, which are functions centered around data points. John and

[2] In a Bayesian interpretation, this assumes a Dirichelet prior $D(2, \ldots, 2)$. More sophisticated estimators are available that incorporate other prior information, but we restrict ourselves to this simple one.

Langley [9] propose to use normal distributions for the kernels where the means are the data points and the variance σ_y is estimated as $1/\sqrt{\#y}$. So, in summary, we have $P(X=x|Y=y)=\sum_{i=1,Y=y}^{n} N(x|x_i,\sigma_y)$.

The benefit of the kernel method is that no assumptions need to be made about the distribution of variable X, any distribution can be approximated and that it is known to have desirable asymptotic properties [9]. On the down side, the method requires storing each of the variable values for each of the continuous variables, so it may require considerable memory for large datasets. Furthermore, calculating $P(X=x|Y=y)$ requires summing over up to n terms, so classification time increases with increasing data sets.

2.3 Discretization method

The *discretization method* converts a continuous variable X into a discrete variable X' after which the distribution $P(X'|Y)$ is estimated using (1). Discretization requires determining the number of values k'_x of the discretized variable X' and finding the k'_x-1 boundary values. The particular choice of k'_x and boundary values can have a dramatic effect on the performance of the classifier [13].

We consider the discretization method based on the minimum description length (MDL) method of Fayyad and Irani [7] since it is widely available and used and is presumed to perform reasonably well compared to other methods [6]. Each of the attributes with continuous variables is considered separately. If the value of X is missing values, the record is ignored, but for ease of exposition we assume that there are no missing values.

The method recursively splits intervals, starting with a single interval containing all data in D. To decide whether it is useful to split an interval D_E containing a set of instances in D into two subintervals D_1 and D_2, the information gain is calculated, which is the difference in entropy of the data in the original interval and the sum of entropies of the subintervals weighed by the number of instances in those intervals. So, the gain is $E.H(D_E)-E_1.H(D_1)-E_2.H(D_2)$ where E, E_1 and E_2 are the size of D_E, D_1 and D_2 respectively. Given an interval D_E, we find the split points that divides D_E into D_1 and D_2 with the highest gain. To accept the division, the gain must exceed a threshold that represents the encoding of the split point and description of classes in the interval. The value of this threshold is $\log_2(n-1)+\log_2(3^{k'_x}-2)-k'_xE+k_1E_1+k_2E_2$ where k_1 and k_2 the number of classes found in the upper and lower subinterval respectively (remember k'_x is the number of classes in interval D_E). See [7] for the exact justification of all the terms.

3 Selecting Classifiers that Perform Well

The classic method of selecting a learning algorithm that performs well is to estimate its performance. We consider three popular ways of estimating accuracy, namely, accuracy measured on the training data, average accuracy of a v-fold cross validation experiment and average accuracy on a repeated v-fold cross validation experiment.

The *test on training set* method works by training the three algorithms on the data set D and then measure how accurate each of the three algorithms is by counting the number of instances in D where the class value y_i is equal to the predicted value $\hat{y}_i$ given the attribute values $\mathbf{x}_i$. The algorithm with the highest count is selected. Benefit of this method is that it does not require a lot of training time, only at most three times as much as learning one of the naive Bayes classifiers. Since all three methods are fast, the overhead in absolute training time is not very large. However, testing on the training data comes with a large cost. In general, the accuracy of a classifier on unseen data will be smaller than on the training data. For example, the nearest neighbor algorithm may be 100% accurate on the training data if there is no noise, however, that does not guarantee good performance on new unseen cases.

A method that suffers less from this over estimation of accuracy is v-fold cross validation. The data is split in v approximately equal parts[3] $D_1, D_2, \ldots, D_v$. An algorithm is trained on all data except the first 'fold' $D \backslash D_1$ and then accuracy is measured on D_1. This process is repeated for each of the folds and the average of the accuracies measured is taken as an estimate of the accuracy of the classifier. The best algorithm according to the estimated accuracy is then trained on the complete data set D and returned. In general, this method returns less biased estimates than measuring on the training set [10].

However, when comparing two algorithms based on a significance test using v-fold cross validation, replicability is very low [2, 3]. This indicates that it is very well possible that sometimes one method has higher accuracy than the other, while in a different random split into the v folds it may be the other way around.

To diminish the sensitivity of the accuracy estimate on the particular randomization into the v folds, we can repeat the v-fold cross validation a number of times and take the average over the results. In a pairwise learning algorithm comparison, this set up tends to increase replicability considerably [2, 3], so we expect it to return more accurate estimates. Like for plain v-fold cross validation, most accurate algorithm is trained on the complete data set D. Repeated cross validation may come at the cost of considerably computational effort compared with the test on training set method when implemented naively.

Naive implementation of (repeated) v-fold cross validation can require training the three algorithms v times (repeatedly). However, the structure of the naive Bayes classifies can be exploited to reduce this effort, as we point out in the following sections.

[3] By stratification on the class, slightly higher accuracy may be obtained, so that is the method used in experiments.

3.1 Cross Validation for the Normal method

Consider the mean m_y^j for fold j for $P(X = x|Y = y)$ with the normal method, which by definition is

$$m_y^j = \frac{1}{\#y_j} \sum_{i=1, i \notin D_j, y_i = y}^{n} x_i$$

where $\#y_j$ is the number of instances in $D \backslash D_j$ for which $Y = y$ and $i \notin D_j$ is short for record i is not in fold D_j. By splitting the sum and multiplying the first part by $\frac{\#y}{\#y}$ we can write

$$m_y^j = \frac{\#y}{\#y_j} \frac{1}{\#y} \sum_{i=1, y_i = y}^{n} x_i - \frac{1}{\#y_j} \sum_{i=1, i \in D_j, y_i = y}^{n} x_i$$

We can write m_y for the mean of X given $Y = y$ in the whole of D and M_y^j for the mean of X given $Y = y$ in fold D_j, that is $M_y^j = \sum_{i=1, i \in D_j, y_i = y}^{n} x_i$. Then, we have

$$m_y^j = \frac{\#y}{\#y_j} m_y - \frac{1}{\#y_j} M_y^j$$

Note that we can calculate m_y and *all* v M_y^js linear in D, since for every instance in D only one of the M_y^j need to be updated.

Likewise, the variance σ_y^j for fold j for $P(X = x|Y = y)$ can be calculated as

$$(\sigma_y^j)^2 = \frac{1}{\#y_j - 1} \sum_{i=1, i \notin D_j, y_i = y}^{n} (x_i^2 - (m_y^j)^2) = \frac{1}{\#y_j - 1} \sum_{i=1, i \notin D_j, y_i = y}^{n} x_i^2 - \frac{\#y_j}{\#y_j - 1} (m_y^j)^2$$

This equals $\frac{\#y-1}{\#y_j-1} \frac{1}{\#y-1} \sum_{i=1, y_i=y}^{n} (x_i^2 - m_y^2) - \frac{1}{\#y_j-1} \sum_{i=1, i \in D_j, y_i=y}^{n} (x_i^2 - m_y^2) + \frac{\#y}{\#y_j-1} m_y^2 - \frac{\#y_j}{\#y_j-1} (m_y^j)^2$. Writing σ_y for the variance on the whole data set D and $S_y^j = \sum_{i=1, i \in D_j, y_i=y}^{n} (x_i^2 - m_y^2) + \frac{\#y}{\#y_j-1} m_y^2 - \frac{\#y_j}{\#y_j-1} (m_y^j)^2$ we get

$$(\sigma_y^j)^2 = \frac{\#y - 1}{\#y_j - 1} \sigma_y - \frac{1}{\#y_j - 1} S_y^j$$

Note that again all terms in can be calculated by one sweep through D. So, performing a single v-fold cross validation experiment using the normal methods can be done in time linear in D. Furthermore, we have m_y and σ_y before we start evaluating so we do not need to retrain if it turns out that the normal method is the best of the methods. Note that the discrete attributes can be handled similarly, as described in Section 3.3.

3.2 Cross Validation for the Kernel method

For the kernel method, classifying an instance in fold D_j requires us for a continuous variable X to calculate $P(X = x|Y = y) = \sum_{i=1, i \notin D_j, Y=y}^{n} N(x|x_i, \sigma_y^j)$ where $\sigma_y^j = 1/\sqrt{\#y_j}$. All the class frequencies $\#y_j$ can be calculated linearly in the size of D and classification likewise. So, performing a v fold cross validation experiment has the same complexity as the test on train method.

3.3 Cross Validation for the Discretization method

The discretization method requires finding the discretization boundaries for each of the v data sets, so a pure implementation needs to run v times. However, if we would discretize the data set D and then run v fold cross validation, we can exploit the following property (the same holds for all discrete variables in the normal and kernel method). $P(X=x|Y=y)$ is estimated using (1), which for experiment on fold j denoted as $P_j(X=x|Y=y)$ becomes

$$P_j(X=x|Y=y)=\frac{1+\sum_{i=1,i\notin D_j,y_i=y,x_i=x}^{n}1}{k_x+\sum_{i=1,i\notin D_j,y_i=y}^{n}1}$$

Note that $\sum_{i=1,i\notin D_j,y_i=y}^{n}1=\#y_j$ and $\sum_{i=1,i\notin D_j,y_i=y,x_i=x}^{n}1=\sum_{i=1,y_i=y,x_i=x}^{n}1-\sum_{i=1,i\in D_j,y_i=y,x_i=x}^{n}1$. Now, denoting $\#(\bar{xy})_j=\sum_{i=1,i\in D_j,y_i=y,x_i=x}^{n}1$

$$P_j(X=x|Y=y)=\frac{P(X=x|Y=y).(k_x+\#y)-\#(\bar{xy})_j}{k_x+\#y_j}$$

where the terms $P(X=x|Y=y)$, $\#y$, all $\#(\bar{xy})_j$, and all $\#y_j$ can all be calculated linearly in D at the cost of some extra memory for storing the terms $\#y$, $\#(\bar{xy})_j$, and $\#y_j$. So, a v fold cross validation experiment can be performed with a discretized data set with the same order of computational effort as the best on train method.

4 Experiments

Taking 25 UCI datasets with continuous variables [1] (properties summarized in table 1) as distributed with Weka [12], we compared naive Bayes using normal distributions with normal kernel distributions [9] and supervised discretization [7]. All experiments were performed with Weka [12] using 10 times 10 fold cross validation since this guarantees high replicability [3]. Algorithms are pairwise compared both with uncorrected and variance corrected paired t-tests [11]. The uncorrected outcomes are presented because most papers use it, which makes our experiments comparable to others. However, uncorrected tests tend to have a Type I error an order of magnitudes larger than the desired significance level while variance corrected tests have a Type I error close to the significance level [3, 11].

4.1 Experiments on naive Bayes methods

Table 2 shows results for the three naive Bayes methods with markers to indicate significant differences. Markers are only added comparing algorithms to the left, so the 'vv' in the kernel method column next to anneal indicates that the kernel method is significantly more accurate than the normal method, but there is no marker in the normal method column indicating that it performs worse than

Table 1. List of datasets.

Dataset	# Number of attributes			Classes		Number of
	Total	Cont.	Nominal	Number	Distribution	Instances
anneal	38	6	32	6	8/99/684/0/67/40	898
arrythmia	279	206	73	13	245/44/15/15/13/25/ 3/2/9/50/0/0/0/4/5/22	452
autos	25	15	10	7	0/3/22/67/54/32/27	205
balance	4	4	0	3	288/49/288	625
breast-w	9	9	0	2	458/241	699
colic	22	7	15	2	232/136	368
credit-a	15	6	9	2	307/383	690
credit-g	20	7	13	2	700/300	1,000
diabetes	8	8	0	2	500/268	768
ecoli	8	7	1	8	143/77/52/35/20/5/2/2	336
glass	8	9	0	7	70/76/17/0/13/9/29	214
heart-c	13	6	7	5	165/138/0/0/0	303
heart-h	13	6	7	5	188/106/0/0/0	294
heart-s	13	6	7	2	150/120	270
hepatitis	19	6	13	2	32/123	155
hypothyroid	29	7	22	4	3481/194/95/2	3,772
ionosphere	34	34	0	2	126/225	351
iris	4	4	0	3	50/50/50	150
labor	16	8	8	2	20/37	57
lymph	18	3	15	4	2/81/61/4	148
segment	19	19	0	7	7 * 330	2,310
sick	29	6	23	2	3541/231	3,772
sonar	60	60	0	2	97/111	208
vehicle	18	18	0	4	212/217/218/199	846
vowel	13	10	3	11	11 * 90	990

the kernel method. The last column shows the maximum standard deviation measured on the accuracy of the three methods. Since the standard deviation tends to be very close (with a few exceptions), this is a good indication of the standard deviation for each of the three methods.

Some observations: The normal method performs significantly worse for the majority of data sets than the kernel method. The exceptions are German credit and diabetes, where the normal method outperforms the kernel method significantly. These results confirm the ones in [9].

However, in [9] reported that colic and cleveland heart disease performed significantly better than the kernel method, a result not replicated in our experiment. This can be explained by the experimental method deployed in [9] having low replicability [3], unlike the 10x10 cross validation method we use here.

Likewise, the normal method performs significantly worse for the majority of data sets than the discretization method. However, there are six data sets where the normal method outperforms the discretization method significantly.

Table 2. Average accuracies (on 10x10 cv experiment) for normal, kernel and discretization methods (standard deviation in brackets). Legend: v/* = better/worse than normal, w/- = better/worse than kernel. Single marker indicate significant difference at 5% for uncorrected test, double markers for corrected test as well.

Dataset	normal	kernel	discretize
anneal	86.6 (3.3)	94.5 (2.4) vv	96.0 (2.2) vv ww
arrhythmia	62.4 (7.0)	66.1 (4.4) v	72.2 (5.3) vv ww
autos	57.4 (10.8)	61.3 (12.0) v	65.2 (10.9) vv w
balance-scale	90.5 (1.7)	91.4 (1.3) v	71.6 (4.8) ** - -
wisconsin-breast-cancer	96.1 (2.2)	97.5 (1.7) vv	97.2 (1.7) vv -
horse-colic	78.7 (6.2)	79.2 (6.2)	79.5 (5.8) v
credit-rating	77.9 (4.2)	81.4 (3.8) vv	86.2 (3.8) vv ww
german-credit	75.2 (3.5)	74.4 (3.6) *	75.0 (3.6) w
pima-diabetes	75.8 (5.3)	75.0 (5.0) *	75.3 (4.8)
ecoli	85.5 (5.5)	86.9 (5.1) v	81.9 (4.7) * - -
Glass	49.5 (9.5)	51.1 (7.0) v	71.9 (8.79) vv ww
cleveland-14-heart-diseas	83.3 (7.2)	84.2 (6.8) v	83.5 (6.9)
hungarian-14-heart-diseas	84.0 (6.3)	85.0 (5.6) v	84.2 (6.3) -
heart-statlog	83.6 (6.0)	84.1 (6.0)	82.6 (6.1) * -
hepatitis	83.8 (9.7)	85.2 (9.4) v	84.3 (10.4)
hypothyroid	95.3 (0.7)	95.9 (0.7) vv	98.2 (0.7) vv ww
ionosphere	82.2 (6.1)	91.8 (4.1) vv	89.4 (4.8) vv -
iris	95.5 (5.0)	96.2 (4.9) v	93.3 (5.8) * -
labor	93.6 (10.2)	93.4 (10.6)	88.6 (13.2) * -
lymphography	83.1 (8.9)	83.2 (8.8)	85.1 (8.3) v w
segment	80.2 (2.1)	85.8 (1.8) vv	91.2 (1.7) vv ww
sick	92.8 (1.4)	95.8 (1.0) vv	97.1 (0.8) vv ww
sonar	67.7 (8.7)	72.4 (8.9) v	76.7 (9.6) vv w
vehicle	44.7 (4.6)	60.9 (3.7) vv	61.1 (3.5) vv
vowel	62.9 (4.4)	70.3 (4.9) vv	58.6 (5.3) ** - -

On the whole, one can conclude that discretization often helps, so the claims to this effect [6, 15] are confirmed. It is noteworthy that these six data sets differ from the two where the normal method outperforms the kernel method.

Though both the kernel method and discretization method outperform the normal method for the majority of the data sets, there is a large performance difference between the two methods. For example, on balance scale the accuracy for the kernel method is 91.44 (1.3) while for the discretization method it is 71.56 (4.77) which is significantly worse. The vowel data shows a similar difference. On the other hand, for the segment data, the kernel method gives 85.77 (1.82) while the discretization method gives 91.15 (1.72). So, surprisingly there are large differences between the performance of those two methods.

Table 3. Average accuracies on 10x10 cross validation experiment for naive Bayes selection algorithms. Legend: b/o = better/worse than normal, v/* = better/worse than kernel, w/- = better/worse than discretization, c/n = better/worse than best on training, d/e = better/worse than best on 10 cv. Single marker indicate significant difference at 5% for uncorrected test, double markers for corrected test as well.

Dataset	best on train	best on 10 cv	best on 10x10 cv	best in literature
anneal	96.0 ± 2.2 bb vv	95.9 ± 2.2 bb vv	95.9 ± 2.2 bb vv	98.2 [13]
arrhythmia	66.0 ± 5.4 b - -	72.2 ± 5.3 bb vv cc	72.2 ± 5.3 bb vv cc	n/a
autos	62.3 ±11.5 b -	63.7 ±12.3 b v - c	65.0 ±11.1 bb v c d	n/a
balance-scale	91.4 ± 1.3 b ww	91.4 ± 1.4 b ww	91.4 ± 1.3 b ww	n/a
breast-wisconsin	97.5 ± 1.7 bb w	97.5 ± 1.7 bb w	97.4 ± 1.8 bb * w n	97.5 [13]
horse-colic	79.3 ± 5.8	79.3 ± 5.9 b	79.4 ± 5.9 b	81.0±2.5[6]
credit-rating	86.2 ± 3.8 bb vv	86.2 ± 3.8 bb vv	86.2 ± 3.8 bb vv	85.9 [13]
german-credit	74.4 ± 3.9 o -	74.8 ± 3.5 o v	74.9 ± 3.6 o v c	75.6±0.9 [6]
pima-diabetes	74.8 ± 5.0 o -	74.9 ± 5.0 o	74.5 ± 5.2 o	76.2±4.8 [8]
ecoli	86.6 ± 5.1 b ww	86.3 ± 5.1 b * ww	86.5 ± 5.2 b * ww	84.0 [13]
Glass	71.9 ± 8.7 bb vv	71.9 ± 8.7 bb vv	71.9 ± 8.7 bb vv	71.5±1.9 [6]
heart-cleveland	84.1 ± 6.8 b	83.2 ± 6.9 * n	83.6 ± 7.1 * n	84.7 [13]
heart-hungarian	84.6 ± 5.9 *	84.4 ± 5.8 *	84.7 ± 5.8 b *	84.1±2.8 [5]
heart-statlog	83.5 ± 6.3 * w	82.8 ± 5.9 o * n	82.9 ± 6.0 * n	n/a
hepatitis	84.7 ± 9.4 b	84.3 ± 9.6 *	84.5 ± 9.6 * c	86.6 [13]
hypothyroid	98.2 ± 0.7 bb vv	98.2 ± 0.7 bb vv	98.2 ± 0.7 bb vv	98.6±0.4 [6]
ionosphere	91.8 ± 4.1 bb w	91.8 ± 4.1 bb w	91.8 ± 4.1 bb w	91.5 [13]
iris	95.9 ± 4.9 w	95.6 ± 5.2 * w	96 ± 4.9 w	96.0±0.3 [9]
labor	93.0 ±10.9 w	92.9 ±11.7 w	93.4 ±10.8 w	94.7 [13]
lymphography	83.3 ± 8.5 -	84.0 ± 8.2 b -	84.4 ± 8.3 b v -	81.6±5.9 [5]
segment	91.2 ± 1.7 bb vv	91.2 ± 1.7 bb vv	91.2 ± 1.7 bb vv	n/a
sick	97.1 ± 0.8 bb vv	97.1 ± 0.8 bb vv	97.1 ± 0.8 bb vv	95.6±0.6 [6]
sonar	76.5 ± 9.3 bb v	76.4 ± 9.4 bb v	76.4 ± 9.4 bb v	77.2 [13]
vehicle	61.0 ± 3.6 bb	60.9 ± 3.4 bb	60.8 ± 3.3 bb	62.3±2.2 [8]
vowel	70.3 ± 4.9 bb ww	70.3 ± 4.9 bb ww	70.3 ± 4.9 bb ww	64.8 [13]

4.2 Experiments on Selection Methods

Table 3 shows results on a 10x10 cross validation experiment[4] for the three methods of selecting one of the methods for handling continuous variables. Markers are shown comparing these methods with the three naive Bayes methods as well. Further, markers are only added comparing algorithms in columns to the left.

One might expect a method that selects the best of the methods to produce a accuracy that is equal to one of the two methods. However, in our experiments, the accuracy of the 'best of both' methods typically differs from the methods

[4] Do not confuse the 10x10 cross validation (cv) experiment with the best on 10x10 cv selector. Note that for the best on 10x10 cv selector in this 10x10 cv experiment, each naive Bayes method is applied 10,000 times on each of the data sets.

Table 4. Ranking of naive Bayes classifiers.

Method	Uncorrected			Corrected		
	total	wins	losses	total	wins	losses
best on 10x10 cv	29	41	12	24	24	0
best on 10 cv	22	36	14	23	23	0
best on train	22	37	15	17	20	3
C4.5						
discretization	-4	33	37	6	20	14
kernel method	1	42	41	-15	12	27
normal method	-70	15	85	-55	2	57

it selects from. The reason is that the 'best of both' methods do not always consistently select the same method for different runs and folds in the 10x10 cross validation experiment. Consequently, the reported accuracy is an average over a mix of the two methods.

Over all, the best on train selector performs remarkably better than the simple methods. It only performs significantly worse (corrected test) than the discretization method on the arrhytmia data set, but otherwise performs equal or significantly better (corrected test) than the normal, kernel and discretization method. On its own, this is already an indication that it helps to be selective about the method for handling continuous variables.

The 10 cv and 10x10 cv selectors perform not significantly worse than the simple methods on any data set but better on many data sets (corrected test). Further, they perform significantly better than the best on train on the arrhytmia data set (corrected). The 10 cv and 10x10 cv selectors perform comparably well, though there is some weak evidence in favor of 10x10, which performs better on the autos set and more often outperforms the other methods.

The last column in Table 3 shows the best results reported in the literature [5, 6, 8, 9, 13] with a wide range of methods for dealing with continuous variables for naive Bayes classifiers. Methods used are 20 times 66%/33% resampling [5], 5 fold cross validation [6, 8], 10 fold cross validation [9], and 10 times 3 fold cross validation [13]. So, mutual comparison of the accuracies should be taken with caution. For example, all but the last method are known to have low replicability [3] and given that they are the best of the reported experiments, should be interpreted as somewhat optimistic compared with our 10 times 10 fold cross validation experiment. Remarkably, all of the best results reported are worse or within the standard deviation of the selector algorithms (except anneal), but never are more than 2.3% better. This suggests that our method at least does not perform worse than the methods reported in the literature.

4.3 Summary of Experiments

Table 4 shows the ranking of the algorithms considered in this paper where the column 'wins' shows the number of data sets where a method is significantly better than an other, 'losses' where it is worse and 'total' is the difference between these two. The algorithms are ranked according to the total of the corrected tests (note that the uncorrected test gives the same ranking except that the kernel and discretization would be swapped). The repeated ten fold cross validation selector stands out considerably according to the uncorrected test and only slightly according to the corrected test. All methods that perform some form of selection perform considerably better than just the pure method. Since 10 fold cross validation can be performed in (about) the same time as best on training selection, this is the method of choice if computation is an issue. Otherwise, repeated cross validation for selecting the method for dealing with continuous variables is recommended.

We compared the naive Bayes methods with C4.5 on the 25 data sets. C4.5 performs significantly better (corrected) than any of the naive Bayes on some data, for instance, vowel and segment. However, in the ranking with corrected tests, C4.5 ended between the simple methods and the selector methods (as shown in Table 4). So, over all the the simple methods perform worse than C4.5, while selector methods perform better.

Furthermore, we performed experiments selecting the best out of the three naive Bayes methods with C4.5, or nearest neighbor or both. Space prevents us to present all results here, but in summary the selection methods never consistently outperformed all classifiers selected among (so sometimes got worse than selection among the naive Bayes methods without C4.5 and/or nearest neighbor). This indicates that simply adding methods to select from does not necessarily increase performance.

5 Conclusions

The contributions of this paper are the following. In this work, we compared all three methods mutually for the first time as far as we know. We used an experimental design that does not suffer from the flaws of the previous empirical comparisons. This comparison shows that all the three methods have their strengths and weaknesses, and none of the three methods systematically outperforms the others on all problems that we considered. We provided a methodology for selecting the best of the three methods and showed how to perform efficient cross validation using the methods. We gave empirical evidence that the method consistently performs at least as good as (according to a 10x10 cv experiment with corrected t-test) any of the other methods on its own. This is remarkable, since selection among naive Bayes together with other methods (C4.5 and nearest neighbor) does not consistently result in the best classifier. Finally, our method is over all better than C4.5 and often appears to perform better than the best naive Bayes classifier reported in the literature.

We recommend that 10 times repeated 10 fold cross validation is used to select a method for dealing with continuous variables. However, if this is computationally impractical, a 10 fold cross validation selection can give reasonable results while being able to be performed in (almost) the same time as selecting the best method on training data.

Work has been done to explain why discretization works for naive Bayes classifiers [8, 15]. This work raises a new question: why does selection of continuous variable handling methods work for naive Bayes classifiers? We suspect that cross validation works well for naive Bayes classifiers because naive Bayes is a stable classifier, that is, it is not very sensitive to leaving samples out of the data set. John and Langley [9] already showed that the learning rate of the normal method can be slightly better than that of the kernel method if the generating distribution is indeed Gaussian. We expect something similar to hold with respect to discretization, as our experiments show that none of kernel and discretization methods uniformly outperforms the others. Conditions under which any of the three methods excel is one of the open questions we would like to address in the future.

There are various directions we would like to explore in the future. Despite the good overall performance of the naive Bayes classifier, it can be outperformed if the basic assumption that all attributes are conditionally independent given the class variable is severely violated. Several methods try to add dependencies in order to overcome this weakness, for example tree augmented naive Bayes, lazy Bayesian rules, and locally weighted naive Bayes. It will be interesting to see whether those methods can benefit from cross validated selection of methods for dealing with continuous variables as well. Preliminary experiments with locally weighted naive Bayes show promising results.

In this paper, we considered selecting the best method for *all* continuous variables. It is very well possible that some variables would be better modeled using a normal distribution and other using a kernel distribution. Not only can variables considered on their own, but even each variable conditioned on each class value. Since this would cause an explosion of possible combinations of distributions smart search criterions should be used. Alternatively, the repeated cross validation criterion could be replaced by a goodness of fit test for each of the continuous variables separately.

References

1. C.L. Blake and C.J. Merz. UCI Repository of machine learning databases. Irvine, CA: University of California, 1998.
2. R.R. Bouckaert. Choosing between two learning algorithms based on calibrated tests. ICML, 51–58, 2003.
3. R.R. Bouckaert and E. Frank. Evaluating the Replicability of Significance tests for comparing learning algorithms. PAKDD, 2004.
4. T.G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. Neural Computation, 10(7) 1895–1924, 1998.
5. P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning, 29, 103–130, 1997.
6. J. Dougherty, R. Kohavi and M. Sahami. Supervised and unsupervised discretization of continuous features. ICML, 194–202, 1995.

7. U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuousvalued attributes for classification learning. IJCAI, 1022–1027, 1993.
8. C.N. Hsu, H.J. Huang and T.T. Wong. Why Discretization Works for Naive Bayes Classifiers. ICML, 399-406, 2000.
9. G.H. John and P. Langley. Estimating Continuous Distributions in Bayesian Classifiers. UAI, 338–345, 1995.
10. R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings IJCAI, 1137–1143, 1995.
11. C. Nadeau and Y. Bengio. Inference for the generalization error. NIPS, 2000.
12. I.H. Witten and E. Frank. Data mining: Practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, San Francisco, 2000.
13. Y. Yang and G.I. Webb. A Comparative Study of Discretization Methods for Naive-Bayes Classifiers. In Proceedings of PKAW 2002, 159-173, 2002.
14. Y. Yang and G.I. Webb. Discretization For Naive-Bayes Learning: Managing Discretization Bias And Variance. Technical Report 2003/131, School of Computer Science and Software Engineering, Monash University. 2003.
15. Y. Yang and G.I. Webb. On Why Discretization Works for Naive-Bayes Classifiers. In Proceedings of the 16th Australian Conference on AI (AI 03), 440-452, 2003.