2011 11th IEEE International Conference on Data Mining

Calculating Feature Weights in Naive Bayes with Kullback-Leibler Measure

Chang-Hwan Lee
Department of Information and Communications
DongGuk University
Seoul, Korea
Email: chlee@dgu.ac.kr

Fernando Gutierrez Dejing Dou
Department of Computer and Information Science
University of Oregon
Eugene, OR, USA
Email: fernando@cs.uoregon.edu dou@cs.uoregon.edu

***Abstract*—Naive Bayesian learning has been popular in data mining applications. However, the performance of naive Bayesian learning is sometimes poor due to the unrealistic assumption that all features are equally important and independent given the class value. Therefore, it is widely known that the performance of naive Bayesian learning can be improved by mitigating this assumption, and many enhancements to the basic naive Bayesian learning have been proposed to resolve this problem including feature selection and feature weighting. In this paper, we propose a new method for calculating the weights of features in naive Bayesian learning using Kullback-Leibler measure. Empirical results are presented comparing this new feature weighting method with some other methods for a number of datasets.**

***Keywords*-Classification; Naive Bayes; Feature Weighting;**

I. Introduction

The naive Bayesian algorithm is one of the most common classification algorithms, and many researchers have studied the theoretical and empirical results of this approach. It has been widely used in many data mining applications, and performs surprisingly well on many applications [2]. However, due to the assumption that all features are equally important in naive Bayesian learning, the predictions estimated by naive Bayesian are sometimes poor. For example, for the problem of predicting whether a patient has a diabetes, his/her *blood pressure* is supposed to be much more important than his/her *height*. Therefore, it is widely known that the performance of naive Bayesian learning can be improved by mitigating this assumption that features are equally important given the class value [12].

Many enhancements to the basic naive Bayesian algorithm have been proposed to resolve this problem. The first approach is to combine feature subset selection with naive Bayesian learning. It is to combine naive Bayesian with a preprocessing step that eliminates redundant features from the data. These methods usually adopt a heuristic search in the space of feature subsets. Since the number of distinct feature subsets grows exponentially, it is not reasonable to do an exhaustive search to find optimal feature subsets.

The second approach is feature weighting method which assigns a weight to each feature in naive Bayesian model. Feature weighting methods are related to a feature subset selection. While feature selection methods assign 0/1 values as the weights of features, feature weighting is more flexible than feature subset selection by assigning continuous weights.

Even though there have been many feature weighting methods, most of them have been applied in the domain of nearest neighbor algorithms [15], and have significantly improved the performance of nearest neighbor methods.

On the other hand, combining feature weighting with naive Bayesian learning received relatively less attention, and there have been only a few methods for combining feature weighting with naive Bayesian learning [6] [7]. The feature weighting methods in naive Bayesian are known to be able to improve the performance of classification learning.

In this paper we propose a feature weighting method for naive Bayesian learning using information theory. The amount of information a certain feature gives to target feature is defined as the importance of the feature, which is measured by using Kullback-Leibler measure. The Kullback-Leibler measure is modified and improved in a number of ways, and the final form eventually serves as the weight of feature. The performance of the proposed method is compared with those of other methods.

The rest of this paper is structured as follows. In Section II, we describe the basic concepts of weighted naive Bayesian learning. Section III shows the related work on feature weighting in naive Bayesian learning, and Section IV discusses the mechanisms of the new feature weighting method. Section V shows the experimental results of the proposed method, and Section VI summarizes the contributions made in this paper.

II. Background

The naive Bayesian classifier is a straightforward and widely used method for supervised learning. It is one of the fastest learning algorithms, and can deal with any number of features or classes. Despite of its simplicity in model, naive Bayesian performs surprisingly well in a variety of problems. Furthermore, naive Bayesian learning is robust enough that small amount of noise does not perturb the results.

In classification learning, a classifier assigns a class label to a new instance. The naive Bayesian learning uses Bayes

1550-4786/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDM.2011.29

theorem to calculate the most likely class label of the new instance. Assume that $a_1, a_2, \cdots, a_n$ are feature values of a new instance. Let C be the target feature which represents the class value, and c represents the value that C can take. A new instance d is classified to the class with the maximum posterior probability. More precisely, the classification on d is defined as follows

$$\mathcal{V}_{map}(d) = \text{argmax}_c \ P(c)P(a_1, a_2, \cdots, a_n|c)$$

In naive Bayesian learning, since all features are considered to be independent given the class value,

$$P(a_1, a_2, \cdots, a_n|c) = \prod_{i=1}^{n} P(a_i|c)$$

Therefore, the maximum posterior classification in naive Bayes is given as

$$\mathcal{V}_{nb}(d) = \text{argmax}_c \ P(c) \prod_{i=1}^{n} P(a_i|c)$$

Since the assumption that all features are equally important hardly holds true in real world application, there have been some attempts to relax this assumption in naive Bayesian learning. The first approach for relaxing this assumption is to select feature subsets in data. In the literature, it is known that the predictive accuracy of naive Bayes can be improved by removing redundant or highly correlated features. This makes sense as these features violate the assumption that each feature is independent on each other. Applying feature subset selection to naive Bayesian learning can be formalized as follows.

$$\mathcal{V}_{fsnb}(d, I(i)) \ = \text{argmax}_c \ P(c) \prod_{i=1}^{n} P(a_i|c)^{I(i)} \quad (1)$$
$$\text{where } I(i) \in \{0, 1\}$$

The feature weighting in naive Bayesian approach is another approach for relaxing the independence assumption. Feature weighting assigns a continuous value weight to each feature, and is thus a more flexible method than feature selection. Therefore, feature selection can be regarded as a special case of feature weighting where the weight value is restricted to have only 0 or 1. The naive Bayesian classification with feature weighting is now represented as follows

$$\mathcal{V}_{fwnb}(d, w(i)) \ = \text{argmax}_c \ P(c) \prod_{i=1}^{n} P(a_i|c)^{w(i)} \quad (2)$$
$$\text{where } w(i) \in \mathbb{R}^+$$

In this formula, unlike traditional naive Bayesian approach, each feature i has its own weight $w(i)$. The $w(i)$ can be any positive number, representing the significance of feature i. Since feature weighting is a generalization of feature selection, it involves a much larger search space than feature selection.

III. Related Work

Feature weighting can be viewed as learning bias, and many feature weighting methods have been applied mostly to nearest neighbor algorithms [15]. While there have been many research for assigning feature weights in the context of nearest neighbor algorithms, very little work of weighting features is done in naive Bayesian learning.

The methods for calculating feature weights can be roughly divided into two categories: filter methods and wrapper methods [10]. These methods are distinguished based on the interaction between the feature weighting and the classification. In filter methods, the bias is pre-determined in advance, and the method calculates incorporate bias as a preprocessing step. Filters are data driven and weights are assigned based on some property or heuristic measure of the data.

In case of wrapper (feedback) method, the performance feedback from the classification algorithm is incorporated in determining feature weights. Wrappers are hypothesis driven. They assign some values to weight vector, and compare the performance of a learning algorithm with different weight vector. In wrapper methods, the weights of features are determined by how well the specific feature settings perform in classification learning. The algorithms iteratively adjust feature weights based on its performance.

An example of wrapper methods for subset selection in naive Bayesian is Langley and Sage's selective Bayesian classifier [12]. They proposed an algorithm called selective Bayesian classifier (SBC) which uses a forward and backward greedy search method to find a feature subset from the whole space of entire features. It uses the accuracy of naive Bayesian on the training data to evaluate feature subsets, and considers adding each unselected feature which can improve the accuracy on each iteration. The method demonstrates significant improvement over naive Bayes.

Another approach for extending naive Bayes is to select a feature subset in which features are conditionally independent. Kohavi [9] presents a model which combines a decision tree with naive Bayes (NBTree). In an NBTree, a local naive Bayes is deployed on each leaf of a traditional decision tree, and an instance is classified using the local naive Bayes on the leaf into which it falls. The NBTree shows better performance than naive Bayes in accuracy.

Hall [7] proposed a feature weighting algorithm using decision tree. This method estimates the degree of feature dependency by constructing unpruned decision trees and looking at the depth at which features are tested in the tree. A bagging procedure is used to stabilize the estimates. Features that do not appear in the decision trees receive a weight of zero. They show that using feature weights with naive Bayes improves the quality of the model compared to standard naive Bayes.

Similar approach is proposed in [14], where they proposed a Selective Bayesian classifier that simply uses only those

features that C4.5 would use in its decision tree when learning a small example of a training set. They present experimental results that this method of feature selection leads to improved performance of the Naive Bayesian Classifier, especially in the domains where naive Bayes performs not as well as C4.5.

Another approach is to extend the structure of naive Bayes to represent dependencies among features. Friedman and Goldszmidt [5] proposed an algorithm called Tree Augmented Naive Bayes (TAN). They assumed that the relationship among features is only tree structure, in which the class node directly points to all feature nodes and each feature has only one parent node from another feature. TAN showed significant improvement in accuracy compared to naive Bayes.

Gartner [6] employs feature weighting performed by SVM. The algorithm looks for an optimal hyperplane that separates two classes in given space, and the weights determining the hyperplane can be interpreted as feature weights in the naive Bayes classifier. The weights are optimized such that the danger of overfitting is reduced. It can solve the binary classification problems and the feature weight is based on conditional independence. They showed that the method compares favorably to state-of-the-art machine learning approaches.

IV. Feature Weighting Method

Weighting features is a relaxation of the assumption that each feature has the same importance with respect to the target concept. Assigning a proper weight to each feature is a process for estimating how much important the feature has.

There have been a number of methods for feature weighting in naive Bayesian learning. Among them, we will use an information-theoretic filter method for assigning weights to features. We choose the information theoretic method because not only it is one of the most widely used methods in feature weighting, but it has strong theoretical background for deciding and calculating weight values.

In order to calculate the weight for each feature, we first assume that when a certain feature value is observed, it gives a certain amount of information to the target feature. In this paper, the amount of information that a certain feature value contains is defined as the discrepancy between prior and posterior distributions of the target feature.

The critical part now is how to define or select a proper measure which can correctly measure the amount of information. Information gain is a widely used method for calculating the importance of features, including decision tree algorithms. It is quite intuitive argument that a feature with higher information gain deserves higher weight.

We note that CN2, a rule induction algorithm developed by [1], minimizes $H(C|A=a)$ in order to search for good rules–this ignores any a priori belief pertaining to C, where H represents the entropy value of C given the observation $A=a$. It assigns the entropy of a posteriori distribution to each inductive rule, and assumes that the rule with higher value of $H(C|A=a)$ is a stronger rule. However, because it takes into consideration only posterior probabilities, this is not sufficient for defining a general goodness measure for weights.

Quinlan proposed a classification algorithm C4.5 [13], which introduces the concept of information gain. The C4.5 uses the information theory that underpins the criterion to construct the decision tree for classifying objects. It calculates the difference between the entropy of a priori distribution and that of a posteriori distribution of class, and uses the value as the metric for deciding branching node. The information gain used in C4.5 is defined as follows.

$$H(C) - H(C|A) = \sum_a P(a) \sum_c P(c|a) \log P(c|a) - \sum_c P(c) \log P(c) \quad (3)$$

Equation (3) represents the discriminative power of a feature and this can be regarded as the weight of a feature.

Since we need the discriminative power of a feature value, we cannot directly use Equation (3) as the measure of discriminative power of a feature value.

In this paper, let us define $\mathcal{IG}(C|a)$ as the instantaneous information that the event $A=a$ provides about C, i.e., the information gain that we receive about C given that $A=a$ is observed. The $\mathcal{IG}(C|a)$ is the difference between a priori and a posteriori entropies of C given the observation a, and is defined as

$$\begin{aligned} \mathcal{IG}(C|a) &= H(C) - H(C|a) \\ &= \sum_c P(c|a) \log P(c|a) - \sum_c P(c) \log P(c) \end{aligned} \quad (4)$$

While the information gain used in C4.5 is information content of a specific feature, the information gain defined in Equation (4) is that of a specific observed value.

However, although $\mathcal{IG}(C|a)$ is a well known formula and uses a more improved measure than CN2, there is a fundamental problem with using $\mathcal{IG}(C|a)$ as the measure of value weight. The first problem is that $\mathcal{IG}(C|a)$ can be zero even if $P(c|a) \neq P(c)$ for some c. For instance, consider the case of an n-valued feature where a particular value of $C=c$ is particularly likely a priori ($p(c) = 1-\epsilon$), while all other values in C are equally unlikely with probability $\epsilon/n-1$. As for $\mathcal{IG}(C|a)$, it can not distinguish the permutation of these probabilities, i.e., an observation which predicts the relatively rare event $C=c$. Since it cannot distinguish between particular events, $\mathcal{IG}(C|a)$ would yield zero information for such events. The following example illustrates this problem in detail.

> **Example 1 :** Suppose the *Gender* feature has values of *m(male)* and *f(female)*, and the target

feature C has the value of y and n. Suppose their corresponding probabilities are given as $\{p(y) = 0.9,\ p(n) = 0.1,\ p(y|m) = 0.1,\ p(n|m) = 0.9\}$ If we calculate $\mathcal{IG}(C|m)$, it becomes

$$\begin{aligned}&\mathcal{IG}(C|m)\\ &= (p(y|m)\log p(y|m) + p(n|m)\log p(n|m)) - \\ &\quad (p(y)\log p(y) + p(n)\log p(n))\\ &= (0.1 \cdot \log(0.1) + 0.9 \cdot \log(0.9)) - \\ &\quad (0.9 \cdot \log(0.9) + 0.1 \cdot \log(0.1)) = 0\end{aligned}$$

We see that the value of $\mathcal{IG}(C|m)$ becomes zero even though the event $male$ significantly impacts on the probability distribution of class C. □

Instead of information gain, in this paper, we employ Kullback-Leibler measure. This measure has been widely used in many learning domains since it originally was proposed in [11]. The Kullback-Leibler measure (denoted as $\mathcal{KL}$) for a feature value a is defined as

$$\mathcal{KL}(C|a) = \sum_c P(c|a)\log\left(\frac{P(c|a)}{P(c)}\right)$$

$\mathcal{KL}(C|a)$ is the average mutual information between the events c and a with the expectation taken with respect to a posteriori probability distribution of C. The difference is subtle, yet significant enough that the $\mathcal{KL}(C|a)$ is always non-negative, while the $\mathcal{IG}(C|a)$ may be either negative or positive.

The $\mathcal{KL}(C|a)$ appears in the information theoretic literature under various guises. For instance, it can be viewed as a special case of the cross-entropy or the discrimination, a measure which defines the information theoretic similarity between two probability distributions. In this sense, the $\mathcal{KL}(C|a)$ is a measure of how dissimilar our a priori and a posteriori beliefs are about C–useful feature value imply a high degree of dissimilarity. It can be interpreted as a distance measure where distance corresponds to the amount of divergence between a priori distribution and a posteriori distribution. It becomes zero if and only if both a priori and a posteriori distributions are identical.

Therefore, we employ the $\mathcal{KL}$ measure as a measure of divergence, and the information content of a feature value a_{ij} is calculated with the use of the $\mathcal{KL}$ measure.

$$\mathcal{KL}(C|a_{ij}) = \sum_c P(c|a_{ij})\log\left(\frac{P(c|a_{ij})}{P(c)}\right) \tag{5}$$

where a_{ij} means the j value of the i-th feature in training data.

The weight of a feature can be defined as the weighted average of the $\mathcal{KL}$ measures across the feature values. Therefore, the weight of feature i, denoted as $w_{avg}(i)$, is defined as

$$\begin{aligned}w_{avg}(i) &= \sum_{j|i} \frac{\#(a_{ij})}{N} \cdot \mathcal{KL}(C|a_{ij})\\ &= \sum_{j|i} P(a_{ij}) \cdot \mathcal{KL}(C|a_{ij})\end{aligned} \tag{6}$$

where $\#(a_{ij})$ represents the number of instances that have the value of a_{ij} and the N means the total number of training instances. In this formula, $P(a_{ij})$ means the probability that the feature i has the value of a_{ij}.

Above weight $w_{avg}(i)$ is biased towards feature with many values, and therefore, the number of records associated with each feature value is too small to make any reliable learning. In order to remove this bias, we incorporate the split information as a part of the feature weight. By using similar split information measure used in decision trees such as C4.5, the final form of the feature weight can be defined as

$$w_{split}(i) = \frac{w_{avg}(i)}{split\_info}$$

where

$$split\_info = -\sum_{j|i} P(a_{ij})\log P(a_{ij})$$

If a feature contains a lot of values, its split information will also be large, which in turn reduces the value of $w_{split}(i)$. Finally the feature weights are normalized in order to keep their ranges realistic. The final form of the weight of feature i, denoted as $w(i)$, is defined as

$$w(i) = \frac{1}{Z} \cdot w_{split}(i) = \frac{1}{Z} \cdot \frac{w_{avg}(i)}{split\_info} \tag{7}$$

$$= \frac{\sum_{j|i} P(a_{ij})\mathcal{KL}(C|a_{ij})}{Z \cdot split\_info} \tag{8}$$

$$= \frac{\sum_{j|i} P(a_{ij}) \sum_c P(c|a_{ij})\log\left(\frac{P(c|a_{ij})}{P(c)}\right)}{-Z \cdot \sum_{j|i} P(a_{ij})\log P(a_{ij})} \tag{9}$$

where Z is a normalization constant

$$Z = \frac{1}{n}\sum_i w(i)$$

In this formula, n represents the number of features in training data. In this paper, the normalized version of $w(i)$ (Equation (9)) is given so as to ensure that $\sum_i w(i) = n$. Algorithm 1 shows the algorithm for naive Bayesian classification using the proposed feature weighting method.

Finally, when we calculate feature weights, we need an approximation method to avoid the problem that the denominator of equations (i.e., $P(c)$) being zero. We use Laplace smoothing for calculating probability values, and the

Algorithm 1: Feature_Weight

Input: a_{ij} : the j-th value in i-th feature, N : total number of records, C : the target feature, d : test data, Z : normalization constant

read training data

for each feature i **do**

calculate $\mathcal{KL}(C|a_{ij}) = \sum_{c} P(c|a_{ij}) \log \left(\frac{P(c|a_{ij})}{P(c)} \right)$

calculate $w_{avg}(i) = \sum_{j|i} \frac{\#(a_{ij})}{N} \cdot \mathcal{KL}(C|a_{ij})$

calculate $w(i) = \frac{w_{avg}(i)}{-Z \cdot \sum_{j|i} P(a_{ij}) \log P(a_{ij})}$

end

for each test data d **do**

class value of $d = \text{argmax}_{c \in C} P(c) \prod_{a_{ij} \in d} P(a_{ij}|c)^{w(i)}$

end

Laplace smoothing methods used in this paper are defined as

$$P(a_{ij}|c) = \frac{\#(a_{ij} \wedge c) + 1}{\#(c_{kl}) + |a_i|}, \quad P(c) = \frac{\#(c) + 1}{N + L} \quad \text{and}$$

$$P(c|a_{ij}) = \frac{\#(a_{ij} \wedge c) + 1}{\#(a_{ij}) + L}$$

where L means the number of class values.

V. Experimental Evaluation

In this section, we describe how we conducted the experiments for measuring the performance of feature weighting method, and then present the empirical results.

We selected 25 datasets from the UCI repository [4]. For the case of numeric features, the maximum number of feature values is not known in advance, or becomes infinite number. In addition, the number of data corresponding to each feature value might be very few, which causes overfitting problem. In light of these, assigning a weight to each numeric feature is not a plausible approach. Therefore, the continuous features in datasets are discretized using the method described in [3]. We omit characteristics of the datasets due to lack of space.

To evaluate the performance, we used 10-fold cross validation method. Each dataset is shuffled randomly and then divided into 10 subsets with the same number of instances. The feature weighting method is implemented in two forms: 1) normal feature weighting method (FWNB), 2) feature weighting method without split information (FWNB-NS). The FWNB method is the feature weighting method described in this paper, and the FWNB-NS method means the FWNB method without split information. These feature weighting methods are compared with other classification methods including regular naive Bayesian, TAN [5], NBTree [9], and decision tree [13]. We used Weka [8] software to run these programs.

Table I shows the accuracies and standard deviations of each method. The ♣ symbol means the top accuracy, and the $*$ means the second accuracy. The bottom rows show the numbers of the 1st and 2nd places for the specific method.

As we can see in Table I, the feature weighting method without split information(FWNB-NS) presents the best performance. It showed the highest accuracy on 6 cases out of 25 datasets, and the second highest accuracies in 5 cases. Altogether, FWNB-NS shows top-2 performance on 11 cases out of 25 datasets. The performance of FWNB is quite competitive as well. It showed top-2 performance on 10 cases out of 25 datasets. Altogether, it can be seen that the overall performance of both FWNB and FWNB-NS is superior to other classification methods including naive Bayesian learning. These results indicate that the proposed feature weighting method could improve the performance of the classification task of naive Bayesian.

In terms of pairwise accuracy comparison between FWNB and FWNB-NS, while both methods show similar performance on 7 cases out of 25 datasets, FWNB-NS showed slightly better performance than FWNB. FWNB-NS outperforms FWNB on 12 cases and FWNB method showed better results on 6 cases. Therefore, it seems that the use of split information($split\_info$) sometimes overcompensates the branching effect of features.

VI. Conclusions

In this paper, a new feature weighting method is proposed for naive Bayesian learning. An information-theoretic method for calculating weight of each feature has been developed using Kullback-Leibler measure.

In order to compare the performance of the proposed feature weighting method, the method is tested using a number of datasets. We present two implementations, one uses split information, while the other does not. These methods are compared with other traditional classification methods. Comprehensive experiments validate the effectiveness of the proposed weighting method. As a result, this work suggests that the performance of naive Bayesian learning can be improved even further by using the proposed feature weighting approach.

Acknowledgments

The first author was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) (Grant number: 2009-0079025 and 2011- 0023296), and part of this work was conducted when the first author was visiting at the University of Oregon.

Table I
ACCURACIES OF THE METHODS

dataset	FWNB	FWNB-NS	NB	TAN	NBTree	J48
abalone	∗ 26.1 ± 1.7	♣ 26.2 ± 1.5	26.1 ± 1.9	25.8 ± 2.0	26.1 ± 1.9	24.7 ± 1.6
balance	71.6 ± 4.4	♣ 72.4 ± 7.4	∗ 72.3 ± 3.8	71.0 ± 4.1	70.8 ± 3.9	69.3 ± 3.8
crx	♣ 86.3 ± 2.7	♣ 86.3 ± 3.8	85.4 ± 4.1	85.8 ± 3.7	85.3 ± 3.9	85.1 ± 3.7
cmc	51.5 ± 5.3	52.5 ± 3.6	52.1 ± 3.8	♣ 54.6 ± 3.8	52.3 ± 4.0	∗ 53.8 ± 3.7
dermatology	♣ 98.5 ± 2.7	97.4 ± 3.1	∗ 97.9 ± 2.2	97.3 ± 2.7	97.0 ± 2.9	94.0 ± 3.3
diabetes	∗ 78.0 ± 3.6	∗ 78.0 ± 4.1	77.9 ± 5.2	♣ 78.6 ± 4.2	77.2 ± 4.7	77.3 ± 4.9
echocardiogram	91.9 ± 8.5	91.9 ± 8.5	90.0 ± 14.0	88.8 ± 13.4	∗ 94.1 ± 10.3	♣ 96.4 ± 7.1
haberman	∗ 74.1 ± 8.0	♣ 74.2 ± 8.5	71.6 ± 3.9	71.6 ± 3.9	71.6 ± 3.9	71.2 ± 3.6
hayes-roth	78.6 ± 10.9	∗ 80.3 ± 8.1	77.0 ± 11.6	64.9 ± 10.5	♣ 82.1 ± 8.7	70.0 ± 9.4
ionosphere	89.1 ± 5.4	89.1 ± 4.6	89.1 ± 5.4	♣ 92.7 ± 4.1	∗ 92.5 ± 4.7	89.4 ± 5.1
iris	94.0 ± 7.3	93.9 ± 7.9	♣ 94.6 ± 5.2	∗ 94.2 ± 5.6	∗ 94.2 ± 5.2	93.8 ± 4.8
kr-vs-kp	89.6 ± 1.4	90.8 ± 0.9	87.8 ± 1.7	92.3 ± 1.5	∗ 97.8 ± 2.0	♣ 99.4 ± 0.3
lung cancer	∗ 71.6 ± 35.1	70.0 ± 33.1	60.0 ± 29.6	61.1 ± 30.2	60.5 ± 29.8	♣ 78.1 ± 22.5
lymphography	78.9 ± 9.6	80.4 ± 11.0	∗ 84.5 ± 9.8	♣ 86.8 ± 8.8	81.4 ± 10.2	76.5 ± 10.1
monk-1	39.7 ± 5.7	∗ 41.3 ± 5.1	39.1 ± 6.2	33.7 ± 4.5	37.5 ± 7.1	♣ 44.8 ± 8.0
postoperative	♣ 70.9 ± 13.0	61.2 ± 17.1	66.5 ± 20.8	66.5 ± 10.9	65.1 ± 11.1	∗ 69.6 ± 6.1
promoters	♣ 93.3 ± 7.8	♣ 93.3 ± 6.5	90.5 ± 8.8	81.3 ± 11.1	87.0 ± 12.8	79.0 ± 12.6
spambase	91.2 ± 1.1	91.0 ± 1.4	90.2 ± 1.2	∗ 93.1 ± 1.1	♣ 93.4 ± 1.1	92.9 ± 1.1
spect	∗ 73.7 ± 18.1	♣ 78.7 ± 17.7	∗ 73.7 ± 20.7	72.5 ± 15.1	67.6 ± 14.6	69.6 ± 14.6
splice	94.1 ± 0.5	94.4 ± 1.3	∗ 95.2 ± 1.3	95.0 ± 1.1	♣ 95.4 ± 1.1	94.1 ± 1.2
tae	48.3 ± 19.0	50.9 ± 13.1	∗ 51.8 ± 12.9	49.5 ± 12.5	♣ 53.2 ± 11.8	50.6 ± 11.5
tic-tac-toe	69.9 ± 2.6	69.9 ± 6.5	70.2 ± 6.1	75.8 ± 3.5	∗ 84.1 ± 3.4	♣ 85.5 ± 3.2
vehicle	62.1 ± 5.7	60.9 ± 7.3	61.9 ± 5.5	♣ 73.3 ± 3.5	70.6 ± 3.5	∗ 70.7 ± 3.8
wine	∗ 97.1 ± 2.9	∗ 97.1 ± 4.7	♣ 97.7 ± 2.9	95.8 ± 4.2	96.1 ± 5.0	90.3 ± 7.2
zoo	93.0 ± 9.4	∗ 96.0 ± 5.1	91.0 ± 12.8	♣ 96.6 ± 5.5	94.4 ± 6.5	92.6 ± 7.3
# 1st	4	6	2	6	4	5
# 2nd	6	5	6	2	5	3
Sum	10	11	8	8	9	8

REFERENCES

[1] Peter Clark and Robin Boswell. Rule induction with cn2: some recent improvements. In *EWSL-91: Proceedings of the European working session on learning on Machine learning*, pages 151–163, 1991.

[2] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3), 1997.

[3] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *International Joint Conference on Artificial Intelligence*, pages 1022–1029, 1993.

[4] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[5] Nir Friedman, Dan Geiger, Moises Goldszmidt, G. Provan, P. Langley, and P. Smyth. Bayesian network classifiers. In *Machine Learning*, pages 131–163, 1997.

[6] Thomas Gärtner and Peter A. Flach. Wbcsvm: Weighted bayesian classification based on support vector machines. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[7] Mark Hall. A decision tree-based attribute weighting filter for naive bayes. *Knowledge-Based Systems*, 20(2), 2007.

[8] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software. *An Update; SIGKDD Explorations*, 11(1), 2009.

[9] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Second International Conference on Knoledge Discovery and Data Mining*, 1996.

[10] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[11] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[12] Pat Langley and Stephanie Sage. Induction of selective bayesian classifiers. In *in Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406, 1994.

[13] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[14] C. A. Ratanamahatana and D. Gunopulos. Feature selection for the naive bayesian classifier using decision trees. *Applied Artificial Intelligence*, 17(5-6):475–487, 2003.

[15] Dietrich Wettschereck, David W. Aha, and Takao Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314, 1997.