A Decomposition Of Classes Via Clustering To Explain And Improve Naive Bayes

R. Vilalta[1] and I. Rish[2]

[1] Department of Computer Science
University of Houston
4800 Calhoun Rd., Houston TX 77204-3010, USA
vilalta@cs.uh.edu
[2] IBM T.J. Watson Research Center
19 Skyline Dr., Hawthorne N.Y. 10532, USA
rish@us.ibm.com

Abstract. We propose a method to improve the probability estimates made by Naive Bayes to avoid the effects of poor class conditional probabilities based on product distributions when each class spreads into multiple regions. Our approach is based on applying a clustering algorithm to each subset of examples that belong to the same class, and to consider each cluster as a class of its own. Experiments on 26 real-world datasets show a significant improvement in performance when the class decomposition process is applied, particularly when the mean number of clusters per class is large.

1 Introduction

Probabilistic classifiers constitute a major venue of research in data mining, pattern recognition, and machine learning. Successful applications are found in speech recognition, document classification, and medical diagnosis, among many others. We focus on a popular probabilistic classifier based on the assumption of attribute independence, also known as Naive Bayes; the performance of this simple classifier is unexpectedly often similar to other classifiers unrestrained by the attribute independence assumption. Although the reasons explaining the competitiveness of Naive Bayes remain unclear, several studies have revealed useful information; examples include studies about the conditions for its optimality [2]; its geometric properties [15]; and how the product distribution implied by the independence assumption compares to most other joint distributions with the same set of marginals [5].

This paper reports on a method to improve the performance of Naive Bayes by attending to the distribution of examples in the input-output space. We work on the characterization and transformation of data rather than on the algorithm design. The idea is to transform the data by decomposing each class into clusters; this is useful to avoid the effects of poor class conditional probabilities based on product distributions when each class spreads into multiple regions. In contrast,

most previous work looks at improving the algorithm design alone; examples include adjusting the estimated probabilities [12], improving probability estimates [7, 14], and combining Naive Bayes with other models [8]. Some previous work does transform the data by searching for attribute dependencies to construct new features [4]; our approach differs in that the transformation is made over the class distribution by looking at each cluster as a new class. Our main idea is to augment the number of original classes according to the example distribution to improve the probability estimations made by Naive Bayes.

Our experimental results, obtained using 26 datasets from the University of California at Irvine repository, enable us to provide an explanation for the competitiveness of Naive Bayes in real-world domains. In summary, most domains exhibit a distribution characterized by few clusters per class; a situation where Naive Bayes is known to perform well. In these cases the performance of our proposed approach is almost identical to Naive Bayes. But when a domain is characterized by many clusters per class (on average) the estimation of class-conditional probabilities is biased, and Naive Bayes performs poorly. In these cases our approach can improve the performance of Naive Bayes significantly. The fact that few domains exhibit many clusters per class explains why Naive Bayes often appears at the same level of performance as other (more sophisticated) algorithms.

This paper is organized as follows. Section 2 introduces background information on classification, probabilistic classifiers, and Naive Bayes. Section 3 explains why Naive Bayes is expected to yield poor probability estimates under certain kinds of input-output distributions. Section 4 describes our class decomposition approach to improve and explain the performance of Naive Bayes. Section 5 compares our class-decomposition approach to local learning. Section 6 reports our experimental analysis. Finally, Section 7 gives a summary and discusses future work.

2 Preliminaries

Let $(A_1, A_2, \cdots, A_n)$ be an n-component vector-valued random variable, where each A_i represents an attribute or feature; the space of all possible attribute vectors is called the input space $\mathcal{X}$. Let $\{y_1, y_2, \cdots, y_k\}$ be the possible classes, categories, or states of nature; the space of all possible classes is called the output space $\mathcal{Y}$. A classifier receives as input a set of training examples $T = \{(\mathbf{x}, y)\}$, where $\mathbf{x} = (a_1, a_2, \cdots, a_n)$ is a vector or point of the input space and y is a point of the output space. We assume T consists of independently and identically distributed (i.i.d.) examples obtained according to a fixed but unknown joint probability distribution ϕ in the input-output space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. The outcome of the classifier is a function h (or hypothesis) mapping the input space to the output space, $h : \mathcal{X} \rightarrow \mathcal{Y}$. Function h can then be used to predict the class of previously unseen attribute vectors.

We consider the case where a classifier defines a discriminant function for each class $g_j(\mathbf{x})$, $j = 1, 2, \cdots, k$ and chooses the class corresponding to the discriminant function with highest value (ties are broken arbitrarily):

$$h(\mathbf{x}) = y_m \texttt{ iff } g_m(\mathbf{x}) \geq g_j(\mathbf{x}) \tag{1}$$

In probabilistic classifiers the discriminant functions are the posterior probabilities of a class given the input vector $\mathbf{x}$, $P(y_j|\mathbf{x})$. Using Bayes rule[3]:

$$g_j(\mathbf{x}) = P(y_j|\mathbf{x}) = \frac{P(\mathbf{x}|y_j)P(y_j)}{P(\mathbf{x})} \tag{2}$$

where $P(y_j)$ is the a priori probability of class y_j, $P(\mathbf{x}|y_j)$ is called the likelihood of y_j with respect to $\mathbf{x}$ or the class-conditional probability, and $P(\mathbf{x})$ is the evidence factor [3]. Since the evidence factor $P(\mathbf{x})$ is constant for all classes we can dispense with it. Assuming all attributes are independent given the class yields the following discriminant function used by Naive Bayes:

$$g_j(\mathbf{x}) = P(y_j) \prod_{i}^{n} P(a_i|y_j) \tag{3}$$

where a_i is the value of attribute A_i in vector $\mathbf{x}$. The main idea is to approximate the joint input-output distribution through a product distribution by assuming attribute independence. While this is clearly unrealistic in many real-world applications, experimental results have repeatedly demonstrated that Naive Bayes often performs as well as other algorithms that make no attribute independence assumption. Our goal in this paper is to relate the performance of Naive Bayes to the characteristics of a domain; the derived analysis shows a clear mechanism to improve the performance of this probabilistic classifier.

3 A Perspective View of Naive Bayes

Although the behavior of Naive Bayes has been explained from different perspectives [2, 15, 5], an understanding of the degree of match between different target distributions and the set of assumptions or bias embedded by the algorithm remains unclear. In this section we identify a kind of distributions for which the product approximation of Naive Bayes may result in multiple misclassifications; we name this problem the *class-dispersion problem.*

3.1 Maximum Entropy and Approximating Distributions

We begin by studying the implication behind a product approximation. Our main assumption is that the set of training examples T is drawn from an unknown but fixed probability distribution ϕ that defines $P(x, y)$ for every point in the

[3] We assume features take on discrete values; we then have probability masses, rather than probability densities.

input-output space. Naive Bayes assumes distribution ϕ can be approximated through a product of low order components (i.e., product of marginals) assuming attribute independence given the class (equation 3). The following definitions will be instrumental in characterizing the approximation used by Naive Bayes.

Definition 1. A distribution $\phi_{\max}$ over the input-output space $\mathcal{Z}$ is called a maximum entropy distribution if it assumes equal probabilities over all elements in $\mathcal{Z}$ (i.e., if it corresponds to a uniform distribution over all possible elements in $\mathcal{Z}$). The entropy of $\phi_{\max}$, denoted as $H_{\phi_{\max}}$, is as high as possible:

$$H_{\phi_{\max}} = -\sum_{i=1}^{|\mathcal{Z}|} \frac{1}{|\mathcal{Z}|} \log \frac{1}{|\mathcal{Z}|} = \log|\mathcal{Z}| \tag{4}$$

where $|\mathcal{Z}|$ is the size of the input-output space.

Definition 2. The information contained in a probability distribution ϕ over the input-output space $\mathcal{Z}$, defined as I_ϕ, is the difference between the entropy of the maximum entropy distribution and the actual entropy of ϕ:

$$I_\phi = H_{\phi_{\max}} - H_\phi \tag{5}$$

where H_ϕ is defined as follows

$$H_\phi = -\sum_{i=1}^{|\mathcal{Z}|} P_i \log P_i \tag{6}$$

and each P_i is the probability of element i in Z according to ϕ.

An interpretation of Definition 2 is straightforward: a flat distribution where all elements are assigned equal probabilities carries no information, whereas the more peaked a distribution, the higher the information conveyed by such distribution [9].

We now consider the problem of approximating a true distribution ϕ using an approximation ϕ'. Let us suppose all we know about ϕ is a set of low order component distributions L. All we require from approximation ϕ' is that it must reduce to the same set of low order components in L (i.e., that it can be expressed as function of the low order components in L). Approximating distributions can be categorized by the amount of information they contain. If the approximation is based on the idea of providing the least amount of additional information beyond the set of low order components, then we have a maximum entropy approximating distribution.

Definition 3. Let ϕ be the true distribution over the input-space $\mathcal{Z}$ and let L be a set of low order components to which ϕ can be reduced. An approximating distribution of ϕ with respect to L is called a maximum entropy approximating distribution, denoted as $\phi'_{\max_L}$, if among all distributions ϕ'_L that reduce to the same set of low order components L, $\phi'_{\max_L}$ is the one with maximum entropy or less information:

$$I_{\phi'_{\max_L}} \leq I_{\phi'_L} \tag{7}$$

for all distributions ϕ'_L that reduced to the same set of low order components L.

3.2 The Product Approximation of Naive Bayes

We now return to the product approximation followed by Naive Bayes. It can be shown that a product approximation contains the smallest amount of information (or maximum entropy) of all possible approximations to ϕ that reduce to the same set of low order components. In other words, Naive Bayes is a maximum entropy approximating distribution [9]. We formalize this as follows: Let L be the set of low order components used by Naive Bayes. That is, for every class y_j, let $L = \{P(y_j),\ P(a_1|y_j),\ P(a_2|y_j),\ \cdots,\ P(a_n|y_j)\}$. Let ϕ_L^{NB} be the product approximation corresponding to Naive Bayes, and let ϕ'_L be any other approximation different from Naive Bayes that reduces to the same set of low order components in L. Then irrespective of the nature of ϕ'_L, it is always true that ϕ'_L contains more (or equal) information than ϕ_L^{NB}.

What is the implication behind the product approximation of Naive Bayes? In brief, such approximation tries to reconstruct the true distribution from the set of low order components assuming as little additional information as possible; hence the distribution is maximally *flat*. Naive Bayes displays a *homogeneous* class distribution on all regions of examples for which the set of low order components is identical.

As an illustration, Figure 1-left shows an input-output distribution on two classes: positive ($y_1 = +$) and negative ($y_2 = -$). We assume a two-dimensional space where attribute A_1 takes on three values, and attribute A_2 takes on two values. Since we have equal class proportions ($P(+) = P(-) = \frac{1}{2}$), the classification depends on the likelihoods only. Figure 1-right shows the approximation made by Naive Bayes. The product approximation tends to smooth all probabilities. According to Naive Bayes the distribution is now the same along $A_2 = 1$, with a likelihood ratio in favor of the negative class, and along $A_2 = 2$, with a likelihood ratio in favor of the positive class.

Consider example $\mathbf{x} = (A_1 = 2, A_2 = 1)$ as shown in Figure 1. Bayes (optimal) classifier assigns $\mathbf{x}$ to class positive (Figure 1-left). The situation changes completely for Naive Bayes (Figure 1-right). Since $P(A_1 = 2|+) = P(A_1 = 2|-) = \frac{1}{3}$, the classification for $\mathbf{x}$ hinges on $P(A_2 = 1|y)$ exclusively; Naive Bayes assigns $\mathbf{x}$ to class negative because $P(A_2 = 1|-) = \frac{2}{3} > P(A_2 = 1|+) = \frac{1}{3}$. The mistake incurred by Naive Bayes stems from the assumption behind a maximal entropy distribution. The existence of regions that are class uniform is blurred by Naive Bayes's vision; these regions are simply averaged altogether when projected onto each attribute.

3.3 The Class-Dispersion Problem

The problem we are addressing is characteristic of distributions where clusters of examples that belong to the same class are dispersed throughout the input

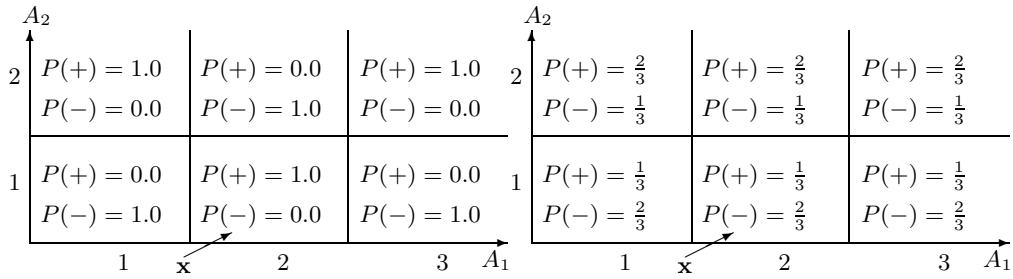

Fig. 1. (left) The true distribution of examples in the input-output space. (right) The maximum-entropy approximation made by Naive Bayes. Example **x** is incorrectly classified by Naive Bayes.

space. We call this the *class-dispersion problem*. In this case, clusters are hard to identify because a single-dimensional projection of the data loses their spatial information. This is related to the small disjunct problem in classification [6], where the existence of many small disjuncts (i.e., class-uniform clusters covering few examples) may account for a significant amount of the total error rate. Our focus, however, is based on the distribution of clusters rather than their coverage.

Our intuition is that Naive Bayes may perform better on domains where the examples of one class are clustered together. This intuition has some theoretical justification. For example, a Boolean target function made of a disjunction (conjunction) of all attributes (or their negations) has only a single example of class 0 (1 for conjunction) and yields optimal (error-free) performance for Naive Bayes [2]. The optimality of Naive Bayes can be easily proven for a more general case of two-class problems where one of the classes is assigned to a single point [11], but the attributes are nominal rather than Boolean. We have extended this result showing that probability distributions having almost all the probability mass concentrated in one example are well approximated through a product distribution (see [11] for proof):

Theorem 1. *If for some* $0 \leq \delta \leq 1$, $\exists\, \mathbf{x}^* = (a_1^*, ..., a_n^*)$ *such that* $P(a_1^*, ..., a_n^*|y_j) \geq 1 - \delta$, *then* $\forall \mathbf{x} = (a_1, ..., a_n)$, $|P(\mathbf{x}|y_j) - \prod_i^n P(a_i|y_j)| \leq n\delta$.

In these cases, although a product approximation does not guarantee good performance of Naive Bayes, it makes it more likely in practice. Nevertheless, as the target distribution changes such that each class groups into multiple clusters, the chances of misclassifications incurred by Naive Bayes increase greatly. This is because Naive Bayes tends to smooth out the class-conditional probabilities. In cases when instances of the same class are scattered, computing marginals (i.e. single-dimensional projections) of the data may result in significant loss of information.

Algorithm 1: Mapping-Process
Input: clustering method C, dataset T
Output: new dataset T'
MAPPING-PROCESS(C,T)
(1) Separate T into subsets $\{T_j\}$
(2) where $T_j = \{(\mathbf{x}, y) \in T | y = y_j\}$
(3) **foreach** T_j
(4) Apply clustering C on T_j
(5) Let $\{C_p^j\}$ be the set of clusters
(6) **foreach** example $e = (\mathbf{x}, y_j)$
(7) Let p be the cluster index for $\mathbf{x}$
(8) Create example $e' = (\mathbf{x}, y'_j)$
(9) where $y'_j = (y_j, p)$
(10) Add e' to T'
(11) **end**
(12) **end**
(13) **return** T'

Fig. 2. The process to transform dataset T into a new dataset T' using a clustering algorithm.

4 Decomposing Classes Into Clusters

Our solution to the class-dispersion problem can be summarized through a two-step process: 1) identify class-uniform clusters of examples in the training set, and 2) relabel each cluster as a new class of examples. The new dataset differs from the original training set in the class labelling: there is now an additional number of classes. Naive Bayes is then trained over the new dataset. During classification, performance can be assessed by simply assigning each example back to its original class. A general description of our approach follows.

4.1 The Data Transformation

Let $T = \{(\mathbf{x}, y)\}$ be the input dataset. Our first step is to map T into another dataset T' through a class-decomposition process. The mapping leaves the input space $\mathcal{X}$ intact but changes the output space $\mathcal{Y}$ into a (possibly) larger space $\mathcal{Y}'$ (i.e., $|\mathcal{Y}'| \geq |\mathcal{Y}|$, where $|\cdot|$ is the cardinality of the space).

The second step is to train Naive Bayes on T' to obtain hypothesis h'. The hypothesis acts over the transformed output space $h' : \mathcal{X} \rightarrow \mathcal{Y}'$. The classification of a new input vector $\mathbf{x}$ is obtained by applying a function g over $h'(x)$ that will essentially bring the class label back to the original output space, $g : \mathcal{Y}' \rightarrow \mathcal{Y}$.

4.2 The Mapping Process

The first step in the transformation process is shown in Algorithm 1 (Figure 2). We proceed by first separating dataset T into sets of examples of the same class. That is T is separated into different sets of examples $T = \{T_j\}$, where each T_j comprises all examples in T labelled with class y_j, $T_j = \{(\mathbf{x}, y) \in T | y = y_j\}$.

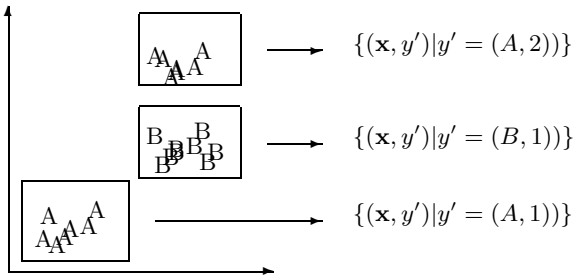

Fig. 3. The mapping process relabels examples to encode both class and cluster.

For each set T_j we apply a clustering algorithm C to find sets of examples (i.e., clusters) grouped together according to some distance metric over the input space. Let $\{C_p^j\}$ be the set of such clusters. We map the set of examples in T_j into a new set T_j' by renaming every class label to indicate not only the class but also the cluster to which each example belongs. One simple way to do this is by making each class label a pair (a, b), where the first element represents the original class and the second element represents the cluster that the example falls into. In that case, $T_j' = \{(\mathbf{x}, y_j')\}$, where $y_j' = (y_j, p)$ whenever example $\mathbf{x}$ is assigned to cluster C_p^j.

An illustration of the transformation above is shown in Figure 3. We assume a two-dimensional input space where examples belong to either class A or B. Let's suppose the clustering algorithm separates class A into two clusters, while class B is grouped into one single cluster. The transformation relabels every example to encode class and cluster label. As a result, dataset T' has now three different classes.

Finally the new dataset T' is simply the union of all sets of examples of the same class relabelled according to the cluster to which each example belongs, $T' = \bigcup_{j=1}^{k} T_j'$.

4.3 The Classification Process

During the second step, Naive Bayes is trained over the new dataset T' producing a hypothesis h' mapping points from input space $\mathcal{X}$ to the new output space $\mathcal{Y}'$. Each discriminant function has the same form as Equation 3, but the number of discriminant functions is now (possibly) larger, according to how much the decomposition process divided up each class into multiple clusters.

When classifying a new input vector $\mathbf{x}$, hypothesis h' will output a prediction consisting of a class label and a cluster label, $h(\mathbf{x}) = (y_j, p)$, corresponding to original class y_j and cluster C_p^j. To know the actual prediction in the original output space $\mathcal{Y}$ we simply apply a function g that removes the second element of the pair, $g(y_j, p) = y_j$. Essentially, we predict class label y_j whenever example $\mathbf{x}$ is assigned to any of the clusters of class y_j.

The decomposition process aims at eliminating the cases where a class spreads out into multiple regions. As each cluster is transformed into a class of its own,

the class-dispersion problem vanishes. The result is a new input-output space where each class sits in a tight region. By reducing the class-dispersion problem, the conditional probabilities estimated by Naive Bayes better conform with the assumption of a product distribution (i.e., of a maximum-entropy distribution).

5 Locality, Capacity, And The Class Decomposition Process

A better understanding of our approach can be gained by looking at the difference between locality, capacity, and the class-decomposition process. Naive Bayes is a global classifier: it makes use of all available data to estimate its parameters (i.e., a priori and class-conditional probabilities). As such it fails to detect local class variations given the same set of low order components. Failing to detect those variations is a byproduct of the attribute-independence assumption; Naive Bayes is a learning machine with low capacity (i.e., low flexibility in the decision boundaries). To solve this problem one may introduce a form of locality in the global classifier, in which parameters are estimated based only on the neighborhood of the example $\mathbf{x}$ being classified.

The class decomposition process discussed in Section 4 introduces an alternative view to local classification: instead of focusing on local regions of the input space, we can augment the number of discriminant functions according to the class distribution. That is, one can add more decision boundaries but retain their low flexibility. By augmenting the number of discriminant functions, the capacity of the algorithm is in fact increased, but the flexibility of the boundaries remains the same. The trick lies in the clustering phase that in fact pre-identifies local structures in the data. In addition, separating classes into clusters simply reduces the dependencies between attributes, but retains all examples for analysis. The class-cluster encoding computed during the transformation process (Figure 3) does not result in a loss of information with respect to the original sample distribution.

6 Experiments

We now report on a series of experiments that compare Naive Bayes (NB) with a modified version (NB′) that computes the transformation described in Section 4. Our datasets (26 domains) can be obtained from the University of California at Irvine repository [1]. In what follows, predictive accuracy on each dataset is obtained using stratified 10-fold cross-validation, averaged over 5 repetitions; tests of significance use a two-tailed t-student distribution. The clustering algorithm follows the Expectation Maximization (EM) technique [10]; it groups examples into clusters by modelling each cluster through a probability density function. Each example in the dataset has a probability of class membership and is assigned to the cluster with highest posterior probability. The number of clusters is estimated using cross-validation. Implementations of Naive Bayes and EM are

Table 1. Predictive accuracy on real-world domains for Naive Bayes with and without the transformation process. Numbers enclosed in parentheses represent standard deviations.

Domain	Naive Bayes NB	Naive Bayes with transformation NB′	Δ Accuracy	Number of Clusters		
				Mean	Min	Max
Anneal	86.64 (0.06)	96.48 (0.09)	9.84*	3.4	1.0	5.0
Audiology	72.17 (0.20)	72.17 (0.20)	0.00	1.0	1.0	1.0
Autos	57.89 (0.25)	71.83 (0.80)	13.94*	2.66	1.0	5.0
Balance-Scale	90.48 (0.05)	90.48 (0.05)	0.00	1.0	1.0	1.0
Breast-Cancer	73.30 (0.15)	73.72 (0.26)	0.42	2.5	2.0	3.0
Breast-W	95.98 (0.01)	95.98 (0.01)	0.00	1.0	1.0	1.0
Colic	78.50 (0.17)	78.50 (0.17)	0.00	1.0	1.0	1.0
Credit-G	75.05 (0.27)	73.26 (0.13)	−1.79	4.0	3.0	5.0
Diabetes	75.49 (0.07)	74.98 (0.10)	−0.51	3.0	1.0	5.0
Heart-C	83.18 (0.07)	84.16 (0.09)	0.98*	2.0	1.0	3.0
Heart-H	84.22 (0.24)	83.52 (0.12)	−0.70	4.0	2.0	6.0
Heart-Statlog	84.28 (0.15)	84.28 (0.15)	0.00	1.0	1.0	1.0
Hepatitis	84.24 (0.25)	85.71 (0.13)	1.47	2.5	1.0	4.0
Ionosphere	82.25 (0.13)	90.12 (0.15)	7.87*	4.5	1.0	8.0
Iris	95.36 (0.07)	95.96 (0.37)	0.60	3.3	2.0	5.0
Chess	87.18 (0.35)	90.62 (0.06)	3.44*	9.5	9.0	10.0
Labor	94.08 (0.42)	94.08 (0.42)	0.00	1.0	1.0	1.0
Lymph	83.79 (0.18)	83.79 (0.18)	0.00	1.0	1.0	1.0
Mushroom	94.01 (0.23)	99.83 (0.01)	5.82*	5.5	5.0	6.0
Tumor	51.20 (0.21)	51.20 (0.21)	0.00	1.0	1.0	1.0
Segment	79.73 (0.09)	87.89 (0.68)	8.16*	4.57	1.0	11.0
Sick	93.84 (0.39)	98.70 (0.04)	4.86*	8.5	6.0	11.0
Vehicle	44.96 (0.17)	73.73 (0.03)	28.77*	8.0	6.0	10.0
Vote	90.07 (0.03)	95.60 (0.13)	5.53*	2.5	2.0	3.0
Vowel	63.79 (0.15)	92.05 (0.11)	28.26*	6.3	4.0	9.0
Zoo	94.92 (0.16)	97.02 (0.00)	2.10*	1.28	1.0	2.0
Average	80.63	85.22	4.58	3.42	2.19	4.80

part of the WEKA machine-learning class library [13], set with default values. Runs were performed on a RISC/6000 IBM model 7043-140.

Table 1 displays our results. The first column describes the domains used for our experiments. The second and third columns report on the accuracy of Naive Bayes; the second column corresponds to the standard version and the third column to the version using the transformation described in Section 4 (numbers enclosed in parentheses represent standard deviations). The fourth column shows the improvement in accuracy that comes with our proposed approach (an asterisk at the top right of each number implies the difference is significant at the $p = 0.01$ level). The last columns shows average values for the mean, minimum, and maximum number of clusters per class for every dataset.

Our results show how the transformation process improves the accuracy of Naive Bayes in most of the datasets used for our experiments. Where no im-

provement is observed the difference is not statistically significant; in the extreme case where each class is grouped into one single cluster, the performance of our proposed approach is identical to Naive Bayes. In some other domains, the improvement goes up to approximately 28% points (e.g., Vehicle and Vowel). The average improvement in accuracy is of approximately 4.5% points. Figure 4 (left) shows the difference between our approach (NB′) and Naive Bayes (NB) (y-axis) where domains are ordered according to the mean number of clusters per class (x-axis). Most significant differences correspond to domains with many clusters per class (we note the increase is not monotonic).

In addition, our results shed some light on the competitiveness of Naive Bayes in real-world domains. Figure 4 (right) shows a histogram of the mean number of clusters per class for each dataset. Most datasets exhibit a distribution characterized by few clusters per class, a situation that favors the assumption behind a product distribution. Few datasets exhibit many clusters per class, which explains why Naive Bayes often appears at the same level of performance as other (more sophisticated) algorithms.

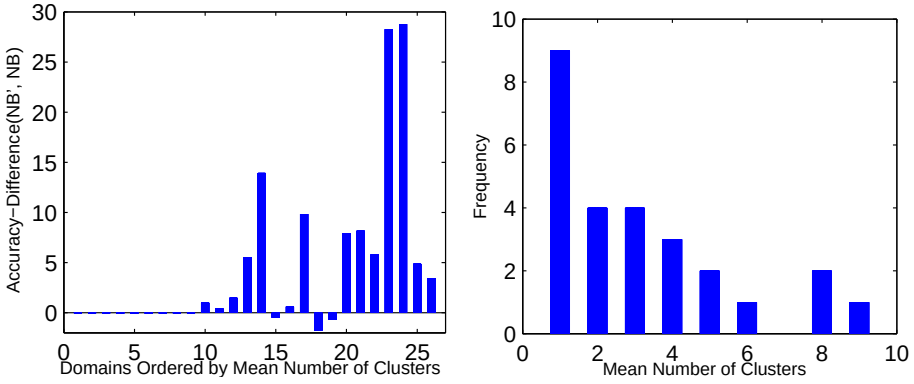

Fig. 4. (left) Accuracy difference between Naive Bayes with the transformation and Naive Bayes standard. (right) A histogram of domains based on the mean number of clusters per class.

7 Summary and Future Work

We propose a method to improve the probability estimates made by Naive Bayes by applying a clustering algorithm to each subset of class-uniform examples; the result is a new output space where each cluster is assigned a new class label. Our experimental analysis shows a significant improvement in performance when the class decomposition process is applied, especially when the mean number of clusters per class is large. The competitiveness of Naive Bayes reported in previous work is explained by the fact that many real-world datasets decompose into few clusters per class, a situation that favors the product distribution assumption followed by Naive Bayes.

Our study assumes an effective clustering algorithm in charge of the class decomposition process. The choice of the clustering algorithm bears relevance to the effectiveness of our approach; future work will explore if our results hold for different clustering algorithms. In addition, we note that the parameters of the clustering algorithm can be adjusted based on the performance of Naive Bayes (e.g., by varying the number of clusters).

Finally, since our proposed approach does not alter the algorithm design, it can be employed outside the boundaries of Naive Bayes, serving as a framework to improve the performance of classifiers that exhibit poor performance when the dataset is characterized by many clusters per class, as is the case with linear classifiers. We plan to address this in future work; our goal is to understand how the class-decomposition process can serve as a general framework to improve classification performance.

References

1. Blake C.L., Merz C.J.: UCI, Repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences (1998). http://www.ics.uci.edu/∼mlearn/MLRepository.html.
2. Domingos P., Pazzani M.: On the Optimality of the Simple Bayesian Classifier Under Zero-One Loss. Machine Learning 29, pp. 103–130 (1997).
3. Duda R. O., Hart P. E., Stork D. G.: Pattern Classification. John Wiley Ed. 2nd Edition (2001).
4. Friedman N., Geiger D., Goldzmidt M.: Bayesian Network Classifiers. Machine Learning 29, pp. 131-163 (1997).
5. Garg A., Roth D.: Understanding Probabilistic Classifiers. European Conference on Machine Learning, Lecture Notes in Artificial Intelligence, pp. 179-191 (2001).
6. Holte R.C., Acker L.E., Porter B.W.: Concept Learning and the Problem of Small Disjuncts. Eleventh International Joint Conference on Artificial Intelligence, Morgan Kaufmann, pp. 813-818 (1989).
7. Kohavi R., Becker B., Sommerfield D.: Improving Simple Bayes. European Conference on Machine Learning (1997).
8. Kohavi R.: Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tee Hybrid. International Conference on Knowledge Discovery and Data Mining (1996).
9. Lewis P.M.: Approximating Probability Distributions to Reduce Storage Requirements. Information and Control, 2, pp. 214-225 (1959).
10. McLachlan G., Krishnan T.: The EM Algorithm and Extensions. John Wiley and Sons (1997).
11. Rish I., Hellerstein, J., Jayram, T.: An Analysis of Naive Bayes on Low-Entropy Distributions. IBM T.J. Watson Research Center, RC91994 (2001).
12. Webb G. I., Pazzani M. J.: Adjusted Probability Naive Bayes Induction. Tenth Australian Joint Conference on Artificial Intelligence. Springer-Verlag, pp. 285-295 (1998).
13. Witten I. H., Frank E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Academic Press, London U.K. (2000).
14. Zadrozny B., Elkan C.: Obtaining Calibrated Probability Estimates From Decision Trees and Naive Bayesian Classifiers. International Conference on Machine Learning (2001).
15. Zhang H., Ling C. X.: Geometric Properties of Naive Bayes in Nominal Domains. European Conference on Machine Learning, pp. 588–599 (2001).