Simultaneous Partitioning of Input and Class Variables for Supervised Classification Problems with Many Classes

Marc Boullé

Abstract In the data preparation phase of data mining, supervised discretization and value grouping methods have numerous applications: interpretation, conditional density estimation, filter selection of input variables, variable recoding for classification methods. These methods usually assume a small number of classes, typically less than ten, and reach their limit in case of too many classes. In this paper, we extend discretization and value grouping methods, based on the partitioning of both the input and class variables. The best joint partitioning is searched by maximizing a Bayesian model selection criterion. We show how to exploit this preprocessing method as a preparation for the naive Bayes classifier. Extensive experiments demonstrate the benefits of the approach in the case of hundreds of classes.

1 Introduction

Supervised classification aims at predicting a class (the value of a target categorical variable) given a set of input numerical or categorical values. Most existing techniques usually consider binary classification or target variables with few classes, typically less than ten. Some applications involve target variables with greater number of classes, such as hand-digit recognition, character recognition or text classification. Recent web-advertising applications have to optimize the choice of a web banner among hundreds in order to maximize the click-through rate given web log data. In case of many classes, the number of instances per class gets smaller, and the reliability of the estimation of class conditional probabilities becomes a problem. In practice, data analysts acknowledge this by restricting to problems with small numbers of classes. Existing methods also assume a small number of classes and are potentially less effective in case of many classes.

Marc Boullé
Orange Labs, 2 avenue Pierre Marzin, 22300 Lannion, France
e-mail: marc.boulle@orange-ftgroup.com

In this paper, we consider the classification problem in its most general setting, without any explicit or implicit assumption related to the number of classes. We focus on univariate data preparation, on the basis of discretization for numerical input variables and value grouping for categorical input variables. These methods have been studied for a long time in the literature [18, 9, 13, 16], as a preprocessing step for decision trees [8, 22, 26] or for naive Bayes classifier [11, 20, 25]. The goal of this paper is to extend data preparation in case of many classes.

Discretization methods split the numerical domain into a set of intervals and grouping methods partition the input categories into groups, in order to estimate the class conditional probabilities. Fine grained partitions allow an accurate discrimination of the classes, whereas coarse grained partitions tend to be more reliable. In case of few classes, these methods provide a robust estimation. In case of many classes, they are either prone to over-fitting or are constrained to under-partition the input variable in order to keep a robust estimation. One way to tackle this problem is to consider the joint partitioning of both the input and class variables. The method introduced in [24] deals with the problem of simultaneously partitioning the row and columns of a contingency table. This method maximizes an association criterion such as Cramer'V, Tschuprow'T or Pearson'ϕ. The heuristic algorithm has a $O(V^5)$ time complexity where V is the maximum number of values (potentially up to the number N of instances), which does not scale in case of variables with many values. In [21], the problem is formalized as that of a block-clustering mixture model and solved using the EM (expectation-maximisation) algorithm. This approach is suitable for exploratory analysis , especially in the case of coclustering of the instances and the variables of a dataset [4]. However, given the computation time requirements, it is not appropriate for data preparation with potentially numerous input variables to preprocess. Among related methods, ECOC (Error-Correcting Output Codes) approaches [10] deal with multi-class classification problems based on the embedding of binary classifiers. The basis of the ECOC approach consists of designing a codeword for each of the classes, which encodes the membership information of each class for a given binary problem. Using multiple codewords, the multi-class problem reduces to a set of binary problems, each based on a bi-partition of the classes. At the decoding step, the set of binary predictions allows to retrieve each individual class. Many coding schemes have been investigated in the literature (one-versus-one, one-versus-all [23], dense random [2],...), as well as decoding designs (Hamming, Euclidean, inverse Hamming, Laplacian,... [12]). Whereas the ECOC approach exploits binary classifiers in case of multi-class problems using predefined bi-partitions of the classes, the purpose of our approach is to improve the accuracy and the reliability of univariate conditional density estimators by searching the most effective partition of the classes for each input variable, which might involve different partitions per input variable.

In this paper, we extend the MODL approach introduced for supervised discretization [6] and value grouping [5]. In this approach, the univariate preprocessing of each input variable is treated as a model selection problem, where a model is defined by a partition of the input values into intervals or group of values, and a multinomial distribution of the classes into each part. The preprocessing model is

extended by simultaneously partitioning the input and class variables and restricting to a multinomial distribution of the groups of classes in each input part. The issue is to optimally balance between accurate models, exploiting many target groups with few classes, and reliable models, based on few groups containing many classes. This problem is tackled and solved using a Bayesian model selection approach to obtain the MAP (maximum a posteriori) model, that is the most probable model given the data.

The paper is organized as follows. Section 2 summarizes the MODL method in the univariate case. Section 3 introduces the extension of the approach with a partitioning of the class variable to tackle the case of many classes. Section 4 presents the impact of this extended preprocessing on the naive Bayes classifier. Section 5 demonstrates the benefits of the approach, both for the data preparation and data modeling phases of data mining. Finally, Section 6 gives a summary.

2 The MODL Supervised Preprocessing Method

This section summarizes the MODL[1] approach in the univariate case, detailed in [6] for supervised discretization, and in [5] for supervised value grouping.

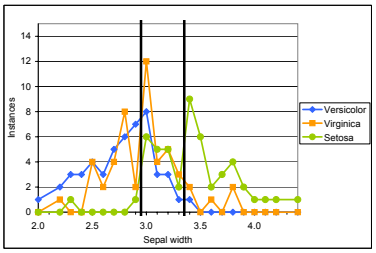
2.1 Discretization

The objective of supervised discretization is to induce a list of intervals which partitions the numerical domain of a continuous input variable, while keeping the information relative to the class variable. A trade-off must be found between information quality (homogeneous intervals in regard to the class variable) and statistical quality (sufficient sample size in every interval to ensure generalization).

Figure 1 illustrates the discretization problem on the Iris dataset [3]. The class variable has three values: Versicolor, Virginica and Setosa. The values of the sepal with input variable are reported on the left, with their frequency per class value. On the right part of figure 1, the input values are discretized into three intervals, which summarize the class conditional density of the input variable.

In the MODL approach, the discretization is turned into a model selection problem. First, a space of discretization models is defined. The parameters of a specific discretization model are the number of intervals, the bounds of the intervals and the frequencies of the classes in each interval. Then, a prior distribution is proposed on this model space. This prior exploits the hierarchy of the parameters: the number of intervals is first chosen, then the bounds of the intervals and finally the frequencies of the classes. The prior is uniform at each stage of the hierarchy. Finally, we assume that the multinomial distributions of the classes in each

[1] Tool available as a shareware at http://perso.rd.francetelecom.fr/boulle/



Total	57	57	36
Versicolor	34	15	1
Virginica	21	24	5
Setosa	2	18	30
	]-∞, 2.95[	[2.95, 3.35[	[3.35, +∞[

Fig. 1 Discretization of the sepal width variable for the classification of the Iris dataset in three classes.

interval are independent from each other. A Bayesian approach is applied to select the best discretization model, which is found by maximizing the probability $p(Model|Data)$ of the model given the data. Using the Bayes rule and since the probability $p(Data)$ is constant under varying the model, this is equivalent to maximizing $p(Model)p(Data|Model)$.

Let N be the number of instances, J the number of classes, I the number of input intervals. $N_{i.}$ denotes the number of instances in the interval i and N_{ij} the number of instances of class j in the interval i. In the context of supervised classification, the number of instances N and the number of classes J are supposed to be known. A discretization model M is then defined by the parameter set $\left\{I, \{N_{i.}\}_{1\leq i\leq I}, \{N_{ij}\}_{1\leq i\leq I, 1\leq j\leq J}\right\}$.

Using the definition of the model space and its prior distribution, Bayes formula can be used to calculate the exact prior probabilities of the models and the probability of the data given a model. Taking the negative log of the probabilities, this provides the evaluation criterion given in Formula 1.

$$\log N + \log \binom{N+I-1}{I-1} + \sum_{i=1}^{I} \log \binom{N_{i.}+J-1}{J-1} + \sum_{i=1}^{I} \log \frac{N_{i.}!}{N_{i1}!N_{i2}!\ldots N_{iJ}!} \quad (1)$$

The first term of the criterion stands for the choice of the number of intervals and the second term for the choice of the bounds of the intervals. The third term corresponds to the parameters of the multinomial distribution of the classes in each interval and the last term represents the conditional likelihood of the data given the model, using a multinomial term. Therefore "complex" models with large numbers of intervals are penalized by the first three terms whereas coarse models are penalized by the last one.

Once the evaluation criterion is established, the problem is to design a search algorithm in order to find a discretization model that minimizes the criterion. In [6], a standard greedy bottom-up heuristic is used to find a good discretization. In order to further improve the quality of the solution, the MODL algorithm performs post-optimizations based on hill-climbing search in the neighborhood of a discretization.

The neighbors of a discretization are defined with combinations of interval splits and interval merges. Overall, the time complexity of the algorithm is $O(JN\log N)$.

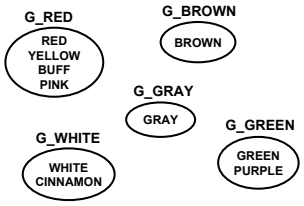
The MODL discretization method for supervised classification provides the most probable discretization given the data. Extensive comparative experiments report high performance [6].

2.2 Value Grouping

Categorical variables are analyzed in a similar way, using a partitioning model of the input values. In the numerical case, the input values are constrained to be adjacent and the only considered partitions are the partitions into intervals. In the categorical case, there are no such constraints between the values and any partition into groups of values is possible. The problem is to improve the reliability of the estimation of the class conditional probabilities owing to a reduced number of groups of values, while keeping the groups as informative as possible. Producing a good grouping is harder with large numbers of input values since the risk of overfitting the data increases. In the extreme situation where the number of values is the same as the number of instances, overfitting is obviously so important that efficient grouping methods should produce one single group, leading to the elimination of the variable.

Figure 2 illustrates the value grouping problem on the Mushroom dataset [3]. The class variable has two values: edible and poisonous. The 10 categorical values of the cap color input variable are reported by decreasing frequency, with their proportion per class value. On the right part of figure 2, the input values are partitioned into 5 groups. For example, the RED and WHITE colors, which have similar proportions of class values, are grouped together.

Value	edible	poisonous	Frequency
BROWN	55.2%	44.8%	1610
GRAY	61.2%	38.8%	1458
RED	40.2%	59.8%	1066
YELLOW	38.4%	61.6%	743
WHITE	69.9%	30.1%	711
BUFF	30.3%	69.7%	122
PINK	39.6%	60.4%	101
CINNAMON	71.0%	29.0%	31
GREEN	100.0%	0.0%	13
PURPLE	100.0%	0.0%	10



Group	edible	poisonous	Frequency
G_RED	38.9%	61.1%	2032
G_BROWN	55.2%	44.8%	1610
G_GRAY	61.2%	38.8%	1458
G_WHITE	69.9%	30.1%	742
G_GREEN	100.0%	0.0%	23

Fig. 2 Value grouping of the categorical values of the cap color variable for the classification of the Mushroom dataset in two classes.

Again, let N be the number of instances, V the number of input values, J the number of classes and I the number of input groups. $N_{i.}$ denotes the number of instances in the group i, and N_{ij} the number of instances of class j in the group i. The Bayesian model selection approach is applied like in the discretization case and provides the evaluation criterion given in Formula 2. This formula has a similar structure as that of Formula 1. The two first terms correspond to the prior distribution of the partitions of the input values, into groups of values in Formula 2 and into intervals in Formula 1. The two last terms are the same in both formula.

$$\log V + \log B(V,I) + \sum_{i=1}^{I} \log \binom{N_{i.}+J-1}{J-1} + \sum_{i=1}^{I} \log \frac{N_{i.}!}{N_{i1}!N_{i2}!\ldots N_{iJ}!} \qquad (2)$$

$B(V,I)$ is the number of divisions of V values into I groups (with eventually empty groups). When $I = V$, $B(V,I)$ is the Bell number. In the general case, $B(V,I)$ can be written as $B(V,I) = \sum_{i=1}^{I} S(V,i)$, where $S(V,i)$ is the Stirling number of the second kind [1], which stands for the number of ways of partitioning a set of V elements into i nonempty sets. In [5], a standard greedy bottom-up heuristic is proposed to find a good partition of the input values. Several pre-optimization and post-optimization steps are incorporated, in order to both ensure an algorithmic time complexity of $O(JN\log(N))$ and obtain accurate value groupings.

3 Extension with Grouping the Class Values

In case of many classes with few instances per class, a reliable estimation of the distribution of the class values is difficult to obtain. We propose to partition the classes into groups of classes, in order to reduce to a more classical supervised classification problem dealing with a small number of groups of classes (kind of "super-classes"), then to describe the true class of each instance given its super-class.

Let Y be a class variable with W classes. The principle of the extended method is to introduce a partition of the W classes into J super-classes. The standard case can be seen as a special case in the extending settings, where $J = W$. W is assumed to be known whereas the number J of super-classes is a parameter that has to be estimated.

Notations:

- N : number of instances
- Y : class variable
- W : number of classes (known)
- J : number of super-classes (unknown)
- $j(w)$: index of the super-class containing class w
- $N_{.j}$: number of instances for super-class j
- m_j : number of classes for super-class j
- n_w : number of instances for class w

For a given number of super-classes J, the goal is to define a partition of the W classes into J super-classes, which amounts to specifying $\{j(w)\}_{1\leq w\leq W}$. Similarly to the case of grouping the values of an input variable reminded in Section 2.2, we use a hierarchical prior for the parameters related to grouping the classes into super-classes:

1. the number of super-classes J is uniformly distributed between 1 and W,
2. for a given number of super-classes W, every division of the W classes into J super-classes is equi-probable.

This corresponds to a prior probability of $\frac{1}{W}$ for the choice of the number of super-classes J. Computing the prior probability for the partition given the number of super-classes is a combinatorial problem similar to that of the grouping problem in Section 2.2, which solution is $\frac{1}{B(W,J)}$. Taking the negative log of these two prior probabilities comes down to introducing the new following prior terms:

$$\log W + \log B(W,J). \tag{3}$$

It is noteworthy that once such a partition is defined, the numbers m_j of classes per super-class can be derived and thus do not belong to the model parameters.

Once the classes are grouped into super-classes, the problem reduces to the standard univariate preprocessing method presented in Section 2. The partitioning models of the input variable are exploited to defined in each input part the local multinomial distribution of the J super-classes. The total number of instances $N_{.j}$ per super-class is calculated using the sum of the local numbers of instances per super-class in each of the I input parts, according to $N_{.j} = \sum_{i=1}^{I} N_{ij}$.

In each super-class, it remains to specify how the instances of the super-class are distributed on the classes. This is done by introducing new modeling parameters, in order to describe locally to each super-class j the multinomial distribution of the $N_{.j}$ instance of the super-class on its m_j classes. As before, a uniform prior is assumed for the parameters of this multinomial model, which comes down to adding the new following prior term:

$$\log \binom{N_{.j} + m_j - 1}{m_j - 1} \tag{4}$$

The likelihood of the multinomial distribution of the instances of each input part on the super-classes is the same as in Section 2 (cf. multinomial term in Formula (1) and (2)). It remains to evaluate the likelihood of the distribution of the instances of each super-class on its classes, using a multinomial term:

$$\log N_{.j}! - \sum_{\{w;j(w)=j\}} \log n_w! \tag{5}$$

Summing these terms on all the super-classes, we obtain:

$$\log W + \log B(W,J) + \sum_{j=1}^{J} \log \binom{N_{.j} + m_j - 1}{m_j - 1} \tag{6}$$

for the prior terms, and

$$\sum_{j=1}^{J} \log N_{.j}! - \sum_{w=1}^{W} \log n_w! \tag{7}$$

for the likelihood terms.

To finish, these new prior and likelihood terms are added to the criterions presented in Section 2. In the case of supervised discretization, Formula (1) is extended to:

$$\begin{aligned} &\log N + \log \binom{N+I-1}{I-1} + \sum_{i=1}^{I} \log \binom{N_{i.}+J+1}{J-1} + \sum_{i=1}^{I} \frac{N_{i.}!}{N_{i1}!N_{i2}!...N_{iJ}!} \\ &+ \log W + \log B(W,J) + \sum_{j=1}^{J} \log \binom{N_{.j}+m_j-1}{m_j-1} + \sum_{j=1}^{J} \log N_{.j}! - \sum_{w=1}^{W} \log n_w! \end{aligned} \tag{8}$$

Similarly, in the case of supervised value grouping, we obtain:

$$\begin{aligned} &\log V + \log B(V,I) + \sum_{i=1}^{I} \log \binom{N_{i.}+J-1}{J-1} + \sum_{i=1}^{I} \log \frac{N_{i.}!}{N_{i1}!N_{i2}!\ldots N_{iJ}!} \\ &+ \log W + \log B(W,J) + \sum_{j=1}^{J} \log \binom{N_{.j}+m_j-1}{m_j-1} + \sum_{j=1}^{J} \log N_{.j}! - \sum_{w=1}^{W} \log n_w! \end{aligned} \tag{9}$$

Optimization Algorithm

The classification problem of simultaneously partitioning one input variable and grouping the values of the class variable is related to parent techniques in the case of regression or bivariate preparation for classification problems. In the case of regression [17], the problem is to simultaneously partition one input variable and discretize the target variable. The case of bivariate preparation for classification involves three variables: two input variables are jointly partitioned in order to discriminate the class variable (the partition of which is not considered). Each problem is specific, leading to significantly different criterions. However, these criterions share a similar additive structure, with terms related to each variable, part, or cell resulting from the cross-product of the univariate partitions. This similar structure allows to reuse the optimization heuristic described in [7].

The main algorithm is a greedy bottom-up merge heuristic, summarized in Algorithm 1. It starts from a random fine-grained bipartition, evaluates each merge between parts of each variable, and performs the best merge while the criterion improves. This main heuristic is enhanced using pre-optimization and post-optimization heuristics, consisting in small perturbations around a current partition (moves of boundaries in case of discretization and moves of values across groups in case of value grouping). Finally, this enhanced greedy heuristic is embedded into the VNS (Variable Neighborhood Search) meta-heuristic [15], which mainly consists of starting from different random partitions (around ten in our experiments).

Algorithm 1 Greedy Bottom-Up Merge heuristic

Require: M {Initial bipartition}
Ensure: $M^*, c(M^*) \leq c(M)$ {Final bipartition with improved cost}
$M^* \leftarrow M$
while improved solution **do**
  $c^* \leftarrow \infty, m^* \leftarrow \emptyset$
  {Evaluate all the merges between adjacent parts of the input variable}
  for all Merge m between two adjacent parts (intervals or groups) of the input variable **do**
    $M' \leftarrow M^* + m$ {Evaluate merge m on bipartition M^*}
    if $c(M') < c^*$ **then**
      $c^* \leftarrow c(M'), m^* \leftarrow m$
    end if
  end for
  {Evaluate all the merges between adjacent groups of the class variable}
  for all Merge m between two adjacent groups of the class variable **do**
    $M' \leftarrow M^* + m$ {Evaluate merge m on bipartition M^*}
    if $c(M') < c^*$ **then**
      $c^* \leftarrow c(M'), m^* \leftarrow m$
    end if
  end for
  {Perform best merge}
  if $c^* < c(M^*)$ **then**
    $M^* \leftarrow M^* + m^*$
  end if
end while

A straightforward implementation of Algorithm 1 leads to a time complexity of $O(N^5)$ where N is the number of instances. However, the method can be optimized in $O(N\sqrt{N}\log N)$ time, as shown in [7]. The optimized algorithm mainly exploits the sparseness of the data, the additivity of the criterion and starts from non-maximal models, refines owing to the pre and post-optimization heuristics. Altogether, this optimization heuristic has a time complexity of $O(N\sqrt{N}\log N)$, whatever be the number of values per variable. The VNS meta-heuristic is exploited to perform anytime optimization: the more you optimize, the better the solution.

4 Impact on the Naive Bayes Classifier

The section recalls the principles of the naive Bayes classifier and describes how to exploit the extended preprocessing introduced in Section 3 to calculate the prediction scores.

4.1 The Naive Bayes Classifier

Let $X = (X_1, X_2, \ldots, X_K)$ be a set of K numerical or categorical input variables and Y a class variable, with W classes $\lambda_1, \lambda_2, \ldots \lambda_W$. Let $x = (x_1, x_2, \ldots, x_K)$ be the input values of a test instance.

The Bayes classifier predicts for each test instance the class with the maximum posterior conditional probability, according to:

$$P(Y = \lambda_w | X = x) = \frac{P(Y = \lambda_w) P(X = x | Y = \lambda_w)}{P(X = x)}.$$

The Bayes classifier is optimal, but it cannot be calculated in practice, since it assumes that the joint class conditional probability is perfectly known. The naive Bayes classifier [19] simplifies the task of estimating the multivariate class conditional probability, using the *naive* assumption that the input variables are independent given the class variable. Also named *idiot's Bayes* in the literature, the naive Bayes classifier performs well in practice on many real datasets [14]. It is easy to implement, fast to train and to deploy, and not prone to overfitting, since the space of models reduces to a singleton. Applying this naive independence assumption, we obtain:

$$P(Y = \lambda_w | X = x) = \frac{P(Y = \lambda_w) \prod_{k=1}^{K} P(X_k = x_k | Y = \lambda_w)}{P(X = x)}. \tag{10}$$

Formula (10) is enough to predict the most probable class given the input values. In applications where a prediction score is necessary, the class conditional probabilities can be calculated by summing over the class terms in the denominator:

$$P(Y = \lambda_w | X = x) = \frac{P(Y = \lambda_w) \prod_{k=1}^{K} P(X_k = x_k | Y = \lambda_w)}{\sum_{v=1}^{W} P(Y = \lambda_v) \prod_{k=1}^{K} P(X_k = x_k | Y = \lambda_v)}. \tag{11}$$

4.2 Using Extended Preprocessing

After the preprocessing step, each variable X_k is partitioned into I_k input parts (intervals or groups of values) for the estimation of the conditional probability of Y, which is itself partitioned into J_k super-classes. Let $N^k_{i_k \cdot}$ be the number of train instances of part i_k of X_k, $N^k_{\cdot j_k}$ that of part j_k of Y and $N^k_{i_k j_k}$ that of cell (i_k, j_k).

Based on the joint partitioning of X_k and Y, let $P^k_{i_k}(x_k)$ be the part related to the input value x_k and $G^k_{j_k}(\lambda_w)$ the super-class related to class λ_w. The preprocessing model provides a piecewise-constant estimation of the conditional probabilities, leading to:

$$P(X_k = x_k | Y = \lambda_w) = P\left(x_k \in P_{i_k}^k(x_k) | \lambda_w \in G_{j_k}^k(\lambda_w)\right),$$
$$P(X_k = x_k | Y = \lambda_w) = \frac{N_{i_k j_k}^k}{N_{\cdot j_k}^k}. \quad (12)$$

The prior probabilities of the classes are estimated using their empirical estimation $P(Y = \lambda_w) = n_w/N$ based on the number n_w of train instances of class λ_w and of the size N of the train sample. Exploiting these empirical probability estimations, Formula (11) turns to:

$$P(Y = \lambda_w | X = x) = \frac{n_w \prod_{k=1}^K \frac{N_{i_k j_k}^k}{N_{\cdot j_k}^k}}{\sum_{v=1}^W n_v \prod_{k=1}^K \frac{N_{i_k j_k}^k}{N_{\cdot j_k}^k}}. \quad (13)$$

In order to avoid zero probabilities, conditional probabilities are estimated using a m-estimate $(support + mp) = (coverage + m)$ with $m = W/N$ et $p = 1/W$.

It is noteworthy that whereas our extended preprocessing method provides an estimation of the conditional probabilities per group of classes (cf. Section 5.1), the naive Bayes classifier combines these coarse grain estimations related to potentially different partitions of the classes, resulting in fine grain estimations per class (cf. Formula 13).

5 Experiments

This section evaluates the impact of our extended preprocessing method on data preparation and on modeling using the naive Bayes classifier.

5.1 Illustrative Example

In order to illustrate the behavior of our method, we use the Letter dataset from the UCI repository [3]. The class value is a capital letter (among 26), that must be identified from a rectangular black-and-white pixel display. The 16 numerical input variables are measures related to the size of the box containing the letter and to statistical moments summarizing the position of the black pixels in the rectangular display. For example, the width of the box is one of these measures. Figure 3 presents a bivariate histogram which displays the results of the simultaneous partitioning of the input *width* variable, into 10 intervals of width, and of the class variable, into 8 groups of letters. The height of the bars stands for the conditional probability of being in a super-class of letters given that the width of the letter belongs to an interval of values. For example, for very small widths ($width <= 0.5$),

the conditional probability of getting the letter I is 100%. On the opposite, for the large widths ($width > 10.5$), the probability of observing a letter in $\{M,W\}$ is 60%, and that of being in $\{X,N,K,H\}$ is 40%. The other cases correspond to intermediate situations. Overall, this preprocessing provides a piecewise-constant estimation of the conditional probabilities, and this estimation is the most probable given the data, according to the Bayesian approach exploited for the selection of the preprocessing model.

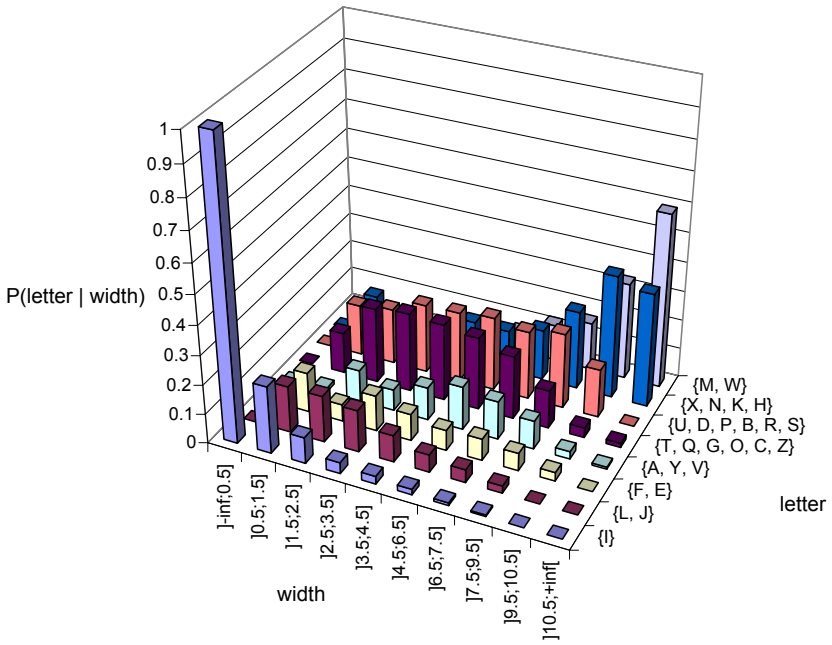


Fig. 3 Estimation of the conditional probability of a letter given its width for the UCI dataset Letter.

5.2 Experiments on the UCI Datasets

In order to assess the benefit of our extended preprocessing method, we evaluate the test accuracy of the naive Bayes classifier using three different preprocessing methods:

- NB(G) simultaneous partitioning of the input and class variables (Section 3),
- NB: partitioning of the input variable only (Section 2),

- nb: standard preprocessing, with ten equal frequency unsupervised discretization for numerical input variables and no value grouping for categorical variables.

The experiments are conducted using 18 datasets of the UCI repository [3], summarized in Table 1. These datasets represent a large variety of domains, numbers of instances (N), numbers of numerical and categorical input variables (K) and have at least three classes (W) with in some cases unbalanced class distribution (Maj. recalls the frequency of the majority class). The test accuracy is evaluated using a stratified 10-fold cross validation. A two-tailed Student test at the 5% confidence level is performed in order to evaluate the significant wins or losses of the NB(G) method versus each other method.

Table 1 Test accuracy on UCI datasets

Dataset	N	K	W	Maj.	NB(G)	NB	nb
Abalone	4177	8	28	0.165	**0.262** $\pm$ 0.022	0.243 $\pm$ 0.028	0.225 $\pm$ 0.020
Flag	194	29	8	0.309	**0.646** $\pm$ 0.083	0.636 $\pm$ 0.070	0.640 $\pm$ 0.088
Glass	214	10	6	0.355	**0.953** $\pm$ 0.046	0.949 $\pm$ 0.048	0.921 $\pm$ 0.042
Iris	150	4	3	0.333	0.913 $\pm$ 0.085	0.920 $\pm$ 0.088	**0.947** $\pm$ 0.040
Led	1000	7	10	0.114	**0.747** $\pm$ 0.038	0.743 $\pm$ 0.032	0.743 $\pm$ 0.032
Led17	10000	24	10	0.107	**0.738** $\pm$ 0.011	0.732 $\pm$ 0.010	0.736 $\pm$ 0.013
Letter	20000	16	26	0.041	**0.747** $\pm$ 0.013	**0.747** $\pm$ 0.013	0.712 $\pm$ 0.012
PenDigits	10992	16	10	0.104	**0.885** $\pm$ 0.012	0.884 $\pm$ 0.010	0.871 $\pm$ 0.010
Phoneme	2254	256	5	0.260	**0.876** $\pm$ 0.023	0.872 $\pm$ 0.025	0.875 $\pm$ 0.023
Satimage	6435	36	6	0.238	0.822 $\pm$ 0.009	**0.823** $\pm$ 0.010	0.812 $\pm$ 0.012
Segmentation	2310	19	7	0.143	0.921 $\pm$ 0.013	**0.923** $\pm$ 0.012	0.899 $\pm$ 0.011
Shuttle	58000	9	7	0.786	0.998 $\pm$ 0.000	**0.999** $\pm$ 0.000	0.992 $\pm$ 0.001
Soybean	376	35	19	0.138	0.918 $\pm$ 0.056	0.926 $\pm$ 0.068	**0.928** $\pm$ 0.060
Thyroid	7200	21	3	0.926	**0.994** $\pm$ 0.002	**0.994** $\pm$ 0.001	0.956 $\pm$ 0.007
Vehicle	846	18	4	0.258	0.595 $\pm$ 0.036	**0.618** $\pm$ 0.031	0.611 $\pm$ 0.035
Waveform	5000	21	3	0.339	**0.811** $\pm$ 0.022	0.810 $\pm$ 0.019	0.808 $\pm$ 0.024
Wine	178	13	3	0.399	**0.983** $\pm$ 0.026	**0.983** $\pm$ 0.026	0.977 $\pm$ 0.028
Yeast	1484	9	10	0.312	**0.575** $\pm$ 0.050	**0.575** $\pm$ 0.046	0.344 $\pm$ 0.032
Mean					0.799	0.799	0.778
W/D/L						2/14/2	8/10/0

Table 1 reports the mean and standard deviation of the test accuracy per dataset, as well as the overall mean on all the datasets (Mean) and the number of wins, draws and losses (W/D/L) of the NB(G) method. The results confirm the significant domination of supervised preprocessing methods, with 8 significants wins and 0 loss of the NB(G) method compared to the standard nb method. However, the results of the NB(G) and NB methods are similar. For UCI datasets with around ten classes, the extended preprocessing method with grouping of the classes is thus interesting for understandability (cf. Section 5.1), but it has no significant impact on test accuracy.

5.3 Experiment with Very Large Number of Classes

In order to study the benefit of our approach in case of very large numbers of classes, we have used the Letter dataset to build a new artificial dataset with many classes. The initial Letter dataset consists of 20000 instances with 16 numerical input variables and 26 classes (alphabet letters). From each pair of instances randomly chosen, we build a new instance that concatenates the two initial instances. We obtain a new bigram dataset containing 10000 instances with 32 input variables and a class variables consisting of 676 bigrams. The average number of instances per class is 15, and the majority class has only 27 instances.

We compare in Table 2 the test accuracy of the NB(G), NB and nb methods using the same stratified 10-fold cross validation protocol as in Section 5.2. As previously, the supervised preprocessing method NB obtains significantly better results than the standard nb method, with 26.7% accuracy against 23.1% accuracy. The extended preprocessing method NB(G) dramatically outperforms the other two methods with 38.9% test accuracy. This clearly demonstrates the benefits of our approach in case of very large numbers of classes.

Table 2 Test accuracy on the bigram dataset.

NB(G)	NB	nb
0.389 $\pm$ 0.016	0.267 $\pm$ 0.016	0.231 $\pm$ 0.016

6 Conclusion

The univariate supervised preprocessing method introduced in this paper exploits a simultaneous partitioning of both the input and class variables. This joint partitioning provides a robust estimation of the class conditional probability, whatever be the number of classes. The best partitioning model is selected using a Bayesian approach and optimized using efficient heuristics with super-linear time complexity. Extensive evaluation on UCI datasets containing around ten classes show that the method obtains test accuracy results with the naive Bayes classifier that are equivalent but not superior to those of the state-of-the-art. On the other hand, for applications with many classes, typically more than one hundred, the experiments demonstrate a significant benefit of our method, with a tremendous increase of the test accuracy. Such dramatic improvements pave the way for a robust, accurate and unified methodology for classification problems, irrespective of the number of classes. In future work, we plan to apply this approach to web advertising problems, where the objective is to choose the banner with the highest predicted click-through rate given user profile data and web page data. Another research direction is to evalu-

ate the benefit of our preprocessing method for alternative classification algorithms, such as decision trees for example.

References

1. Abramowitz, M., Stegun, I.: Handbook of mathematical functions. Dover Publications Inc., New York (1970)
2. Allwein, E., Schapire, R., Singer, Y.: Reducing multiclass to binary: A unifying approach for margin classifiers. Journal of Machine Learning Research **1**, 113–141 (2002)
3. Asuncion, A., Newman, D.: UCI machine learning repository (2007). URL http://www.ics.uci.edu/∼mlearn/MLRepository.html
4. Bock, H.: Simultaneous clustering of objects and variables. In: E. Diday (ed.) Analyse des Données et Informatique, pp. 187–203. INRIA (1979)
5. Boullé, M.: A Bayes optimal approach for partitioning the values of categorical attributes. Journal of Machine Learning Research **6**, 1431–1452 (2005)
6. Boullé, M.: MODL: a Bayes optimal discretization method for continuous attributes. Machine Learning **65**(1), 131–165 (2006)
7. Boullé, M.: Optimum simultaneous discretization with data grid models in supervised classification: a bayesian model selection approach. Advances in Data Analysis and Classification **3**(1), 39–61 (2009)
8. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. California: Wadsworth International (1984)
9. Catlett, J.: On changing continuous attributes into ordered discrete attributes. In: Proceedings of the European Working Session on Learning, pp. 87–102. Springer (1991)
10. Dietterich, T., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research **2**, 263–286 (1995)
11. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: Proceedings of the 12th International Conference on Machine Learning, pp. 194–202. Morgan Kaufmann, San Francisco, CA (1995)
12. Escalera, S., Pujol, O., Radeva, P.: On the decoding process in ternary error-correcting output codes. IEEE Transactions in Pattern Analysis and Machine Intelligence **32**(1), 120–134 (2010)
13. Fayyad, U., Irani, K.: On the handling of continuous-valued attributes in decision tree generation. Machine Learning **8**, 87–102 (1992)
14. Hand, D., Yu, K.: Idiot's bayes ? not so stupid after all? International Statistical Review **69**(3), 385–399 (2001)
15. Hansen, P., Mladenovic, N.: Variable neighborhood search: principles and applications. European Journal of Operational Research **130**, 449–467 (2001)
16. Holte, R.: Very simple classification rules perform well on most commonly used datasets. Machine Learning **11**, 63–90 (1993)
17. Hue, C., Boullé, M.: A new probabilistic approach in rank regression with optimal bayesian partitioning. Journal of Machine Learning Research pp. 2727–2754 (2007)
18. Kass, G.: An exploratory technique for investigating large quantities of categorical data. Applied Statistics **29**(2), 119–127 (1980)
19. Langley, P., Iba, W., Thompson, K.: An analysis of Bayesian classifiers. In: 10th National Conference on Artificial Intelligence, pp. 223–228. AAAI Press (1992)
20. Liu, H., Hussain, F., Tan, C., Dash, M.: Discretization: An enabling technique. Data Mining and Knowledge Discovery **4**(6), 393–423 (2002)
21. Nadif, M., Govaert, G.: Block clustering of contingency table and mixture model. In: Advances in Intelligent Data Analysis VI, *LNCS*, vol. 3646, pp. 249–259. Springer (2005)
22. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
23. Rifkin, R., Klautau, A.: In defense of one-vs-all classification. Journal of Machine Learning Research **5**, 101–141 (2004)

24. Ritschard, G., Zighed, D.A.: Simultaneous row and column partitioning: the scope of a heuristic approach. In: Foundations of Intelligent Systems, *LNAI*, vol. 2871, pp. 468–472. Springer (2003)
25. Yang, Y., Webb, G.: A comparative study of discretization methods for naive-Bayes classifiers. In: Proceedings of the Pacific Rim Knowledge Acquisition Workshop, pp. 159–173 (2002)
26. Zighed, D., Rakotomalala, R.: Graphes d'induction. Hermes, France (2000)