Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Improving naive Bayes classifier by dividing its decision regions*

Zhi-yong YAN, Cong-fu XU†‡, Yun-he PAN

(*Institute of Artificial Intelligence, Zhejiang University, Hangzhou 310027, China*)

†E-mail: xucongfu@zju.edu.cn

Received Dec. 20, 2010; Revision accepted Apr. 8, 2011; Crosschecked July 14, 2011

Abstract: Classification can be regarded as dividing the data space into decision regions separated by decision boundaries. In this paper we analyze decision tree algorithms and the NBTree algorithm from this perspective. Thus, a decision tree can be regarded as a classifier tree, in which each classifier on a non-root node is trained in decision regions of the classifier on the parent node. Meanwhile, the NBTree algorithm, which generates a classifier tree with the C4.5 algorithm and the naive Bayes classifier as the root and leaf classifiers respectively, can also be regarded as training naive Bayes classifiers in decision regions of the C4.5 algorithm. We propose a second division (SD) algorithm and three soft second division (SD-soft) algorithms to train classifiers in decision regions of the naive Bayes classifier. These four novel algorithms all generate two-level classifier trees with the naive Bayes classifier as root classifiers. The SD and three SD-soft algorithms can make good use of both the information contained in instances near decision boundaries, and those that may be ignored by the naive Bayes classifier. Finally, we conduct experiments on 30 data sets from the UC Irvine (UCI) repository. Experiment results show that the SD algorithm can obtain better generalization abilities than the NBTree and the averaged one-dependence estimators (AODE) algorithms when using the C4.5 algorithm and support vector machine (SVM) as leaf classifiers. Further experiments indicate that our three SD-soft algorithms can achieve better generalization abilities than the SD algorithm when argument values are selected appropriately.

Key words: Naive Bayes classifier, Decision region, NBTree, C4.5 algorithm, Support vector machine (SVM)
doi:10.1631/jzus.C1000437 **Document code:** A **CLC number:** TP181

1 Introduction

The naive Bayes classifier (Domingos and Pazzani, 1997) is an example of global learning (Huang *et al.*, 2008), which obtains a distribution estimation of the whole data set. The naive Bayes classifier assumes that attributes of instances are independent given the class (Domingos and Pazzani, 1997). Although this assumption is very naive, the naive Bayes classifier has good generalization ability, and is one of top 10 algorithms in data mining voted by IEEE International Conference on Data Mining (ICDM) 2006 (Wu *et al.*, 2008).

There are many studies on improving the generalization ability of the naive Bayes classifier, amongst which the NBTree algorithm (Kohavi, 1996) is typical. The NBTree algorithm trains naive Bayes classifiers on the leaf nodes of a decision tree.

Some researchers regard classification as dividing data space $\boldsymbol{X}$ into some decision regions separated by decision boundaries (Bishop, 2006), although most researchers regard classification as finding a mapping from data space $\boldsymbol{X}$ to label set $\boldsymbol{Y}$ (Mitchell, 1997). We call the former perspective the dividing perspective. In this paper, we analyze decision tree algorithms and the NBTree algorithm from the dividing perspective. A decision tree can be regarded as a classifier tree composed of two types of classifiers. An NBTree can be regarded as a two-level classifier tree with a decision tree classifier as the root node and several naive Bayes classifiers as leaf nodes. Then the NBTree algorithm can be regarded as training naive Bayes

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 60970081) and the National Basic Research Program (973) of China (No. 2010CB327903)

classifiers in decision regions of the C4.5 algorithm (Quinlan, 1993). The NBTree algorithm can use advantages of both the C4.5 algorithm and the naive Bayes classifier, and outperforms these two classifiers (Kohavi, 1996).

Being contrary to the NBTree algorithm which trains naive Bayes classifiers in decision regions of the C4.5 algorithm, we propose the second division (SD) algorithm to train classifiers in decision regions of the naive Bayes classifier. The SD algorithm will generate a two-level classifier tree, with the naive Bayes classifier as the root node and other classifiers as leaf nodes. We also propose three soft versions of the SD algorithm (SD-soft) to deal with overlapped regions generated by the naive Bayes classifier. In this paper leaf classifiers used are the naive Bayes classifier, the C4.5 algorithm, and support vector machine (SVM) (Vapnik, 1995). We perform experiments on 30 data sets from the UC Irvine (UCI) repository (Frank and Asuncion, 2010) to compare the SD algorithm with the NBTree and the averaged one-dependence estimators (AODE) algorithms (Webb *et al.*, 2005). We also compare three SD-soft algorithms with the SD algorithm. We apply the SD algorithm to the AODE algorithm, the C4.5 algorithm, and SVM. Finally we adopt the global/local learning theory of Huang *et al*. (2008) to discuss why the SD algorithm works.

2 Analysis of decision tree and the NBTree algorithms from the dividing perspective

The dividing perspective regards classifiers as dividing data space $\boldsymbol{X}$ into some decision regions (Bishop, 2006). The definition of the decision region is as follows.

Definition 1 (Decision region) If region $\boldsymbol{R}$ in data space $\boldsymbol{X}$ satisfies the following two conditions for classifier C, then it is called the decision region of label Y_i under classifier C, denoted as $\mathbf{DR}(C, Y_i)$:

$$\forall \boldsymbol{x} \in \boldsymbol{R},\ C(\boldsymbol{x}) = Y_i, \tag{1}$$

$$\forall \boldsymbol{x} \notin \boldsymbol{R},\ C(\boldsymbol{x}) \neq Y_i, \tag{2}$$

where $\boldsymbol{x}$ is the test instance.

For decision regions, the following formulas are true:

$$\mathbf{DR}(C, Y_i) \neq \varnothing, \forall Y_i \in \boldsymbol{Y}, \tag{3}$$

$$\mathbf{DR}(C, Y_i) \cap \mathbf{DR}(C, Y_j) = \varnothing,\ \ Y_i \neq Y_j, \tag{4}$$

$$\bigcup_{Y_i \in \boldsymbol{Y}} \mathbf{DR}(C, Y_i) = \boldsymbol{X}. \tag{5}$$

From Eqs. (3)–(5), it is clear that decision regions of classifier C constitute a partition of data space $\boldsymbol{X}$.

From the dividing perspective, a classifier can be regarded as a divider, whose function is to obtain a partition of data space $\boldsymbol{X}$.

2.1 Analysis of decision tree algorithms

The decision tree is a knowledge representation method. Non-leaf nodes N_i of a decision tree are attributes, and leaf nodes of a decision tree are labels. On each non-leaf node, branches are generated according to value of the attribute of this node. An example of the decision tree is shown in Fig. 1.

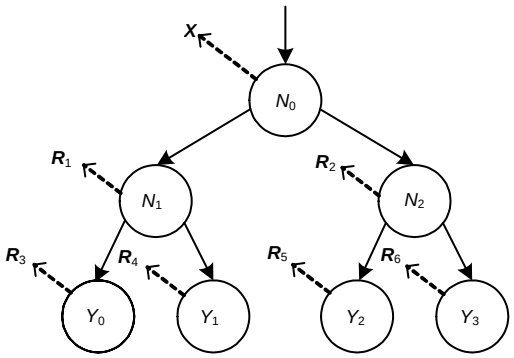

Fig. 1 Structure of a decision tree and regions corresponding to its nodes

Each node of a decision tree corresponds to a region of data space $\boldsymbol{X}$. For example, the root node N_0 of the decision tree in Fig. 1 corresponds to data space $\boldsymbol{X}$. A leaf node of a decision tree corresponds to a part of a decision region. Decision tree algorithms adopt a majority voting method to determine the label for the region corresponding to a leaf node. For each non-leaf node, decision tree algorithms divide its corresponding region into several sub-regions which correspond to its children nodes. The relationship among regions of parent and children nodes of the decision tree in Fig. 1 is as follows:

$$\boldsymbol{X} = \boldsymbol{R}_1 \cup \boldsymbol{R}_2, \boldsymbol{R}_1 \cap \boldsymbol{R}_2 = \varnothing, \boldsymbol{R}_1 \neq \varnothing, \boldsymbol{R}_2 \neq \varnothing, \tag{6}$$

$$\boldsymbol{R}_1 = \boldsymbol{R}_3 \cup \boldsymbol{R}_4, \boldsymbol{R}_3 \cap \boldsymbol{R}_4 = \varnothing, \boldsymbol{R}_3 \neq \varnothing, \boldsymbol{R}_4 \neq \varnothing, \tag{7}$$

$$\boldsymbol{R}_2 = \boldsymbol{R}_5 \cup \boldsymbol{R}_6, \boldsymbol{R}_5 \cap \boldsymbol{R}_6 = \varnothing, \boldsymbol{R}_5 \neq \varnothing, \boldsymbol{R}_6 \neq \varnothing. \tag{8}$$

It is clear that each non-leaf node corresponds to a divider. From the dividing perspective, a classifier can be regarded as a divider. If regions generated by a divider are associated with labels, then regions can be regarded as decision regions. After this process, a divider can be regarded as a classifier. For example, if nodes N_1 and N_2 are associated with labels Y_1 and Y_2, respectively, then the root node N_0 can be regarded as corresponding to a classifier whose output is Y_1 or Y_2. Each non-leaf node corresponds to a piece-wise classifier (PWC), which is a very simple classifier:

$$\mathrm{PWC}(\boldsymbol{x})=\begin{cases}Y_0, & x[i]<v_1,\\ Y_1, & x[i]\in[v_1,v_2),\\ \vdots & \\ Y_{k-1}, & x[i]\geq v_{k-1},\end{cases} \tag{9}$$

where $x[i]$ is the value of the ith attribute of instance $\boldsymbol{x}$ and v_i is the interval boundary of $x[i]$.

The region corresponding to a node can be regarded as a decision region of the classifier corresponding to the parent node. For a non-leaf node, decision tree algorithms train a PWC in the decision region of the classifier corresponding to its parent node. In Fig. 1, PWC_0 corresponding to the root node N_0 divides data space $\boldsymbol{X}$ into $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$, which correspond to nodes N_1 and N_2 respectively. For node N_1, PWC_1 is trained in decision region $\boldsymbol{R}_1$ of PWC_0. Similarly, for node N_2, PWC_2 is trained in decision region $\boldsymbol{R}_2$ of PWC_0.

Each leaf node corresponds to a majority voting classifier (MVC), which is also a very simple classifier:

$$\mathrm{MVC}(\boldsymbol{x})=\arg\max_{Y_i}\left|\left\{(\boldsymbol{x}_j,y_j)\,|\,(\boldsymbol{x}_j,y_j)\in D, y_j=Y_i\right\}\right|, \tag{10}$$

where $\boldsymbol{x}_j$ is the instance and D denotes the training data set.

Decision tree algorithms train an MVC in the region corresponding to a leaf node. The label predicted by a leaf node is the one predicted by the MVC trained in the region corresponding to the leaf node.

Each non-leaf node corresponds to a PWC, and each leaf node corresponds to an MVC; thus, a decision tree can be regarded as a classifier tree. The classifier tree corresponding to the decision tree in Fig. 1 is shown in Fig. 2.

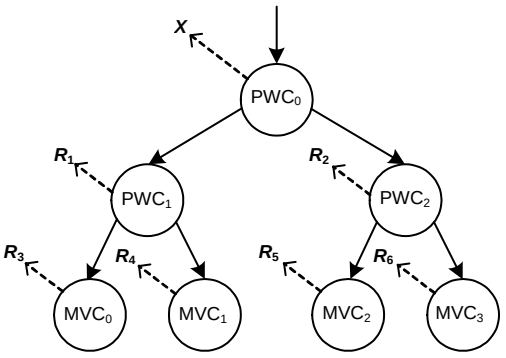

Fig. 2 Classifier tree corresponding to the decision tree in Fig. 1

Although both PWC and MVC are very simple, we can obtain very good generalization ability by organizing these two classifiers as a tree. For example, the C4.5 algorithm is one of top 10 algorithms in data mining voted by ICDM 2006 (Wu *et al*., 2008).

2.2 Analysis of the NBTree algorithm

The NBTree algorithm also generates a decision tree, whose leaf nodes are naive Bayes classifiers instead of labels. An NBTree is also a classifier tree, whose non-leaf and leaf nodes are PWCs and naive Bayes classifiers, respectively. The structure of an NBTree is shown in Fig. 3.

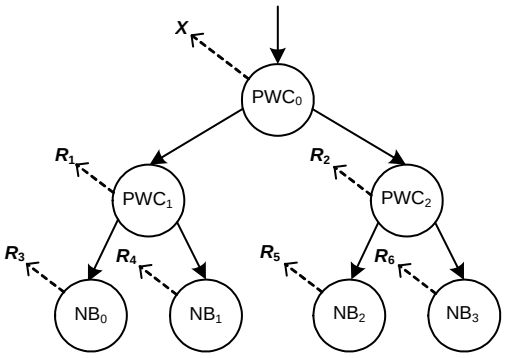

Fig. 3 An NBTree as a classifier tree with naive Bayes (NB) classifiers as leaf nodes

In Fig. 3, if nodes NB_0, NB_1, NB_2, and NB_3 are associated with labels Y_0, Y_1, Y_2, and Y_3, then regions $\boldsymbol{R}_3$, $\boldsymbol{R}_4$, $\boldsymbol{R}_5$, and $\boldsymbol{R}_6$ can be regarded as decision regions of PWC_1 and PWC_2. Then the sub-tree composed of PWC_0, PWC_1, and PWC_2 can be regarded as a decision tree generated by the C4.5 algorithm, whose objective is to generate decision regions for training

naive Bayes classifiers. Thus, an NBTree can also be regarded as a classifier tree composed of a C4.5 classifier and several naive Bayes classifiers. These naive Bayes classifiers are trained in decision regions of the C4.5 classifier. The two-level classifier tree corresponding to the classifier tree in Fig. 3 is shown in Fig. 4.

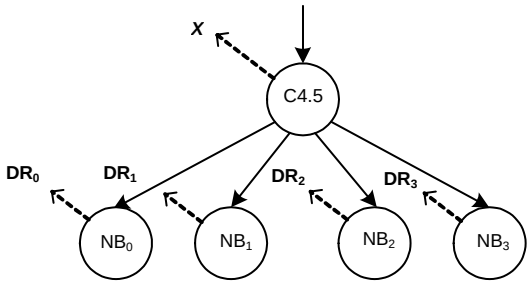

Fig. 4 An NBTree composed of a C4.5 and several naive Bayes classifiers

3 Training classifiers in decision regions of the naive Bayes classifier

Training naive Bayes classifiers in decision regions of the C4.5 algorithm can improve the generalization abilities of both the naive Bayes classifier and the C4.5 algorithm (Kohavi, 1996). In this section, we study the method of training classifiers in decision regions of the naive Bayes classifier.

3.1 SD algorithm

The simplest method is to train classifiers in decision regions of the naive Bayes classifier, which is shown in Fig. 5.

There are two questions when training classifiers in decision regions of the naive Bayes classifier:

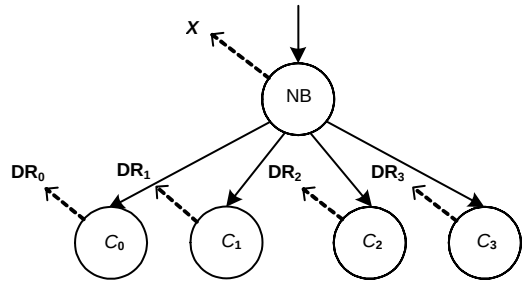

Fig. 5 Classifier training in decision regions of the naive Bayes classifier

1. When should we train classifier C in a decision region of the naive Bayes classifier?

2. When classifier C is trained in a decision region of the naive Bayes classifier, there are two classifiers in this decision region. Then which classifier should be selected?

For question 1, it is clear that when classifier C can improve the generalization ability of the naive Bayes classifier in the decision region, classifier C should be trained. However, it is very difficult to determine whether classifier C can do this. Thus, we exclude only one impossible case when the training accuracy of the naive Bayes classifier in the decision region is 100%. When this case is true, all instances in the decision region have the same class. If classifier C is trained in the decision region, only a classification model with Y_i as its unique output can be obtained. This classification model cannot improve the generalization ability of the naive Bayes classifier in this decision region.

For question 2, between the naive Bayes classifier and classifier C, the one with better generalization ability should be selected, but it is very difficult to determine in the training phase. There are at least three methods for selecting a better classifier. The first method is to divide training data set D into training subset and validation subset and then to choose the classifier with better test accuracy on the validation subset. The second method adopts cross validation to select a better classifier. The third method is to select the classifier with better training accuracy. Amongst above three methods, the former two may obtain more accurate selection, but they simultaneously need more time. We adopt the last method in this study.

Training the classifier can be regarded as dividing data space $\boldsymbol{X}$ once. Then training classifier C in decision regions of the naive Bayes classifier can be regarded as the SD to data space $\boldsymbol{X}$. The procedure of the SD algorithm is shown in Fig. 6. Like the NBTree algorithm, the SD algorithm also generates a two-level classifier tree. In this classifier tree, the naive Bayes classifier is the root classifier, and classifier C is the leaf classifier.

3.2 SD-soft algorithm

The SD algorithm trains classifiers in decision regions of the naive Bayes classifier. Decision regions constitute a partition of data space $\boldsymbol{X}$. This dividing is

Training phase:
Input: training data set D, the naive Bayes classifier NB, and classifier C
Output: a two-level classifier tree named SD tree (composed of NB, C_i, and indicator vector **is_used**)
1. Train the naive Bayes classifier NB on training data set D.
2. Obtain data subsets $\{D_i\}$ in decision regions of NB using the following formula:
$D_i = \{(\boldsymbol{x}, y) \mid \mathrm{NB}(\boldsymbol{x}) = Y_i\}$.
3. For each D_i, compute training accuracy of NB. If it is 100%, is_used[i]=false. If it is not 100%, train C on D_i to obtain C_i. If training accuracy of C_i on D_i is greater than that of NB, then is_used[i]=true; otherwise, is_used[i]=false.

Test phase:
Input: an SD tree, and test instance $\boldsymbol{x}$
Output: label y
1. Obtain prediction y_{NB} of NB for $\boldsymbol{x}$.
2. If is_used[y_{NB}]==false, y=y_{NB}; otherwise, y=$C_{y\mathrm{NB}}(\boldsymbol{x})$.

Fig. 6 Procedure of the SD algorithm

hard dividing. But the naive Bayes classifier can generate overlapped regions in data space $\boldsymbol{X}$, which means instance ($\boldsymbol{x}$, y) may belong to several decision regions with conditional probabilities. In this section we propose a soft version of the SD algorithm to deal with overlapped regions generated by the naive Bayes classifier.

There are many methods for generating regions with overlaps. In this section, we adopt the method similar to the one used in the divide-and-conquer (D&C) algorithm (Frosyniotis *et al*., 2003) to generate overlapped regions and combine classifiers. The D&C algorithm first uses a clustering algorithm to obtain data subsets $\{D_i\}$, and then trains several multi-layered perceptron classifiers (Pal and Mitra, 1992) on these data subsets. The clustering algorithm used is the fuzzy C-means algorithm (Bezdek, 1981) or greedy expectation minimization (EM) algorithm (Vlassis and Likas, 2002). In the test phase, predictions of these classifiers are combined to make the final prediction. The membership degree of instance ($\boldsymbol{x}$, y) belonging to cluster i is denoted as $u(\mathrm{cluster}_i, \boldsymbol{x})$. The D&C algorithm utilizes membership threshold q to obtain soft data subsets. The soft data subset D_{si} is obtained by

$$D_{si} = \left\{(\boldsymbol{x}, y) \mid u(\mathrm{cluster}_i, \boldsymbol{x}) > q\right\}. \tag{11}$$

The D&C algorithm then uses Eq. (12) to obtain the probability distribution $p(Y_i|\boldsymbol{x})$:

$$p(Y_i \mid \boldsymbol{x}) = \sum_{j=0}^{k-1} u(\mathrm{cluster}_j, \boldsymbol{x}) \cdot I(C_j(\boldsymbol{x}) == Y_i). \tag{12}$$

The indicator function $I(z)$ is defined by

$$I(z) = \begin{cases} 1, & \text{if } z \text{ is true,} \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

We adopt the above subset generating method to obtain regions with overlaps. The output of the naive Bayes classifier is a probability distribution vector $[p_{\mathrm{NB}}(Y_0|\boldsymbol{x}), p_{\mathrm{NB}}(Y_1|\boldsymbol{x}), \ldots, p_{\mathrm{NB}}(Y_{k-1}|\boldsymbol{x})]$. The data subset D_{si} is obtained by

$$D_{si} = \{(\boldsymbol{x}, y) \mid p_{\mathrm{NB}}(Y_i \mid \boldsymbol{x}) > q\}. \tag{14}$$

From Eq. (14), it is clear that instance ($\boldsymbol{x}$, y) may be contained in several data subsets.

Since classifier C may be a classifier that outputs the probability distribution, we improve the combining method of the D&C algorithm:

$$p(Y_i \mid \boldsymbol{x}) = \sum_{j=0}^{k-1} p_{\mathrm{NB}}(Y_j \mid \boldsymbol{x}) \cdot p_{C_j}(Y_i \mid \boldsymbol{x}), \tag{15}$$

where $p_{C_j}(Y_i \mid \boldsymbol{x})$ denotes the probability of $\boldsymbol{x}$ belonging to class Y_i predicted by classifier C_j.

Taking indicator vector **is_used** into consideration, the output of $p(Y_i|\boldsymbol{x})$ is calculated by

$$\begin{aligned} p(Y_i \mid \boldsymbol{x}) = &\sum_{j=0}^{k-1} p_{\mathrm{NB}}(Y_j \mid \boldsymbol{x}) \cdot p_{C_j}(Y_i \mid \boldsymbol{x}) \cdot I(j, T) \\ &+ p_{\mathrm{NB}}(Y_i \mid \boldsymbol{x}) \cdot I(i, F), \end{aligned} \tag{16}$$

where $I(j, T)$ and $I(i, F)$ are used to represent I(is_used[j]==true) and I(is_used[i]==false), respectively. Then we have

$$I(j, T) + I(i, F) = 1. \tag{17}$$

Eq. (18) is true for every classifier that outputs the probability distribution:

$$\sum_{i=0}^{k-1} p_C(Y_i \mid \boldsymbol{x}) = 1. \tag{18}$$

Then we have the following derivation:

$$\begin{aligned}\sum_{i=0}^{k-1} p(Y_i \mid \boldsymbol{x}) &= \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} p_{\mathrm{NB}}(Y_j \mid \boldsymbol{x}) \cdot p_{C_j}(Y_i \mid \boldsymbol{x}) \cdot I(j,T) \\ &\quad + \sum_{i=0}^{k-1} p_{\mathrm{NB}}(Y_i \mid \boldsymbol{x}) \cdot I(i,F) \\ &= \sum_{j=0}^{k-1} p_{\mathrm{NB}}(Y_j \mid \boldsymbol{x}) \cdot I(j,T) \cdot \sum_{i=0}^{k-1} p_{C_j}(Y_i \mid \boldsymbol{x}) \\ &\quad + \sum_{i=0}^{k-1} p_{\mathrm{NB}}(Y_i \mid x) \cdot I(i,F) \\ &= \sum_{j=0}^{k-1} p_{\mathrm{NB}}(Y_j \mid \boldsymbol{x}) \cdot I(j,T) \\ &\quad + \sum_{i=0}^{k-1} p_{\mathrm{NB}}(Y_i \mid \boldsymbol{x}) \cdot I(i,F) \\ &= \sum_{i=0}^{k-1} p_{\mathrm{NB}}(Y_i \mid \boldsymbol{x}) \cdot (I(j,T) + I(i,F)) \\ &= \sum_{i=0}^{k-1} p_{\mathrm{NB}}(Y_i \mid \boldsymbol{x}) \\ &= 1.\end{aligned}$$

The final output is determined by

$$y = \arg\max_{Y_i \in \boldsymbol{Y}} p(Y_i \mid \boldsymbol{x}). \tag{19}$$

The SD-soft algorithm is shown in Fig. 7.

Training phase:
Input: training data set D, the naive Bayes classifier NB, classifier C, and probability threshold q.
Output: a two-level classifier tree named SD tree (composed of NB, C_i, and indicator vector **is_used**)
1. Train the naive Bayes classifier NB on training data set D.
2. Obtain soft data sets $\{D_{si}\}$ and data set $\{D_i\}$ in decision regions of NB using the following formulae:
$$D_{si} = \{(\boldsymbol{x}, y) \mid p_{\mathrm{NB}}(Y_i \mid \boldsymbol{x}) > q\},$$
$$D_i = \{(\boldsymbol{x}, y) \mid \mathrm{NB}(\boldsymbol{x}) = Y_i\}.$$
3. For each D_i, compute training accuracy of NB. If it is 100%, is_used[i]=false. If it is not 100%, train C on D_{si} to obtain C_i. If training accuracy of C_i on D_i is greater than that of NB, then **is_used**[i]=true; otherwise, is_used[i]=false.

Test phase:
Input: an SD tree, and test instance $\boldsymbol{x}$
Output: label y
1. Obtain prediction y_{NB} of NB for $\boldsymbol{x}$.
2. If is_used[y_{NB}]==false, $y=y_{\mathrm{NB}}$. Otherwise, obtain the probability distribution of NB for $\boldsymbol{x}$, and then use the following two equations to obtain the final prediction y:
$$p(Y_i \mid \boldsymbol{x}) = \sum_{j=0}^{k-1} p_{\mathrm{NB}}(Y_j \mid \boldsymbol{x}) \cdot p_{C_j}(Y_i \mid \boldsymbol{x}) \cdot I(j,T) + p_{\mathrm{NB}}(Y_i \mid \boldsymbol{x}) \cdot I(i,F),$$
$$y = \arg\max_{Y_i \in Y} p(Y_i \mid \boldsymbol{x}).$$

Fig. 7 Procedure of the SD-soft algorithm

3.3 Time complexity of the SD algorithm

We first assume that the training data has m attributes, n instances, and k classes, and then discuss the time complexity of the SD algorithm. It is clear that the SD algorithm increases the computational cost of the naive Bayes classifier. The training and test time complexities of the naive Bayes classifier are $O(mn)$ and $O(km)$, respectively (Webb *et al.*, 2005). In the training phase, after training the naive Bayes classifier, the SD algorithm first divides the training data set into data subsets, and the time complexity is $O(kmn)$. Then the SD algorithm trains classifier C k times at most, and the time complexity is $O(kC_{\mathrm{train}}(n/k))$, in which $O(C_{\mathrm{train}}(n/k))$ is the time complexity of training classifier C on a data set of size n/k. Thus, the training time complexity of the SD algorithm is $O(mn)+O(kmn)+O(kC_{\mathrm{train}}(n/k))$. In the test phase, the SD algorithm first uses NB to obtain the prediction y, and the time complexity is $O(km)$. Then the SD algorithm may use classifier C_y to make the final prediction, and $O(C_{\mathrm{test}}(n/k))$ is the time complexity of using classifier C trained on the data set with size n/k to make the prediction. Thus, the test time complexity of the SD algorithm is $O(km)+O(C_{\mathrm{test}}(n/k))$.

In this study, we use the naive Bayes classifier, the C4.5 algorithm, and SVM as leaf classifiers, and SVM adopts the sequential minimal optimization (SMO) training algorithm (Platt, 1999). These three classifiers are eager classifiers, whose test time complexity is very small. Thus, we compare only the training time complexity of the SD algorithm with that of the naive Bayes classifiers, the NBTree algorithm, and the AODE algorithm. Training time complexities are listed in Table 1.

From Table 1, it is clear that training time complexities of SD (C=NB) and SD (C=C4.5) are smaller than that of the NBTree algorithm, and the training time complexity of SD (C=SVM) may be comparable with that of the NBTree algorithm. The training time complexity of SD (C=C4.5) may be comparable with

that of the AODE algorithm. The training time complexity of SD (C=SVM) is larger than that of the AODE algorithm. The computational cost of the SD-soft algorithm is larger than that of the SD algorithm because data set D_{si} usually has more instances than data set D_i.

Table 1 Training time complexities of eight classifiers

Classifier	Time complexity	Literature
NB	$O(mn)$	Webb *et al*., 2005
NBTree	$O(n^2km^2/v)$	Zheng and Webb, 2005
AODE	$O(m^2n)$	Webb *et al*., 2005
C4.5	$O(mn\log n)$	Frank and Witten, 1998
SVM	$O(n^d)$, $d\in[1, 2.2]$	Platt, 1999
SD[1]	$O(2mn)+O(kmn)$	This study
SD[2]	$O(mn)+O(kmn)+O(mn\log(n/k))$	This study
SD[3]	$O(mn)+O(kmn)+O(n^d/k^{d-1})$	This study

SD[1]: C=NB; SD[2]: C=C4.5; SD[3]: C=SVM. k: number of classes; m: number of attributes; v: average number of values for an attribute; n: number of training instances

4 Experimental results

In this section, we perform experiments to use the SD and the SD-soft algorithms to train the naive Bayes classifier, the C4.5 algorithm, and SVM in decision regions of the naive Bayes classifier. We first compare the SD-algorithm with the NBTree and the AODE algorithms, and then compare the SD-soft algorithm with the SD algorithm.

The experimental environment is the Waikato environment for knowledge analysis (WEKA) (Witten and Frank, 2005). The naive Bayes classifier, the NBTree algorithm, and the C4.5 algorithm adopt NaiveBayes, NBTree, and J48 methods of WEKA respectively, with default settings. The J48 method is an implementation of the C4.5 release 8 (Quinlan, 1996). The AODE algorithm adopts the AODE method of WEKA, and adopts the supervised discretion method to discretize continual attributes. SVM adopts the SMO method of WEKA, and a linear kernel function.

The experimental data sets are 30 UCI data sets. We remove instances with unknown values from these data sets. After the above preprocessing, descriptions of these data sets are listed in Table 2.

We adopt 10 times 10-fold cross validation to obtain test accuracies and adopt two-tailed t-test with 95% confidence to compare different classifiers to obtain win-loss-tie (W/L/T) records. Due to space limitation, only t-test results are listed.

Table 2 Descriptions of 30 UCI data sets after preprocessing

Data set	Number of attributes	Number of classes	Number of instances
Anneal	39	6	898
Balance-scale	5	3	625
Breast-cancer	10	2	277
Bridges	13	6	70
Car	7	4	1728
Credit-g	21	2	1000
Dermatology	35	6	358
Diabetes	9	2	768
Ecoli	8	8	336
Haberman	4	2	306
Heart-c	14	5	296
Heart-statlog	14	2	270
Hepatitis	20	2	80
Ionosphere	35	2	351
Iris	5	3	150
Liver-disorders	7	2	345
Lung-cancer	57	2	27
Molecular	59	4	106
Mushroom	23	2	5644
Postoperative	9	3	87
Segment	20	7	2310
Solar	13	2	323
Sonar	61	2	208
Soybean	36	19	562
Sponge	46	3	54
Tic-tac-toe	10	2	958
Vehicle	19	4	846
Vote	17	2	232
Vowel	14	11	990
Zoo	18	7	101

4.1 SD algorithm

We first use the naive Bayes classifier as the leaf classifier. The SD algorithm can be applied twice or three times, denoted as 2SD and 3SD, respectively. We train the naive Bayes classifier, SD, 2SD, and 3SD, and then compare these classifiers by t-test (Table 3).

From Table 3, we can find that SD, 2SD, and 3SD all can improve the generalization ability of the naive Bayes classifier.

Compared with the naive Bayes classifier, 2SD obtains the best *t*-test result. Thus, it is necessary to apply the SD algorithm twice.

Table 3 The *t*-test results (W/L/T) of applying the SD algorithm once, twice, and three times

Algorithm	W/L/T	Algorithm	W/L/T
SD vs. NB	14/6/10	2SD vs. SD	3/1/26
2SD vs. NB	14/5/11	3SD vs. 2SD	2/0/28
3SD vs. NB	14/6/10		

Then we compare 2SD with the NBTree and the AODE algorithms (Table 4).

Table 4 shows that the generalization ability of 2SD is better than that of the NBTree algorithm, but worse than that of the AODE algorithm. Another advantage of 2SD over the NBTree algorithm is that the training time of 2SD is much shorter than that of the NBTree algorithm, because the SD algorithm does not adopt cross validation in the training phase.

Table 4 The *t*-test results (W/L/T) of comparing 2SD with NBTree and AODE

Algorithm	W/L/T
2SD vs. NBTree	13/8/9
2SD vs. AODE	8/12/10

We then use the C4.5 algorithm and SVM as leaf classifiers of the SD algorithm. Test accuracies obtained by the SD algorithm are compared with those obtained by the naive Bayes classifier, leaf classifiers, the NBTree algorithm, and the AODE algorithm (Table 5).

Table 5 The *t*-test results (W/L/T) of comparing the SD algorithm with other four classifiers

Algorithm	W/L/T
SD (*C*=C4.5) vs. NB	15/4/11
SD (*C*=C4.5) vs. C4.5	18/2/10
SD (*C*=C4.5) vs. NBTree	18/4/8
SD (*C*=C4.5) vs. AODE	16/6/8
SD (*C*=SVM) vs. NB	16/3/11
SD (*C*=SVM) vs. SVM	15/7/8
SD (*C*=SVM) vs. NBTree	20/1/9
SD (*C*=SVM) vs. AODE	16/6/8

From Table 5, it is clear that when using the C4.5 algorithm and SVM as leaf classifiers, the SD algorithm can obtain better generalization abilities than the naive Bayes classifier, leaf classifiers, the NBTree, and the AODE algorithms.

4.2 SD-soft algorithm

The selection of q value is crucial for the SD-soft algorithm. In our experiment, the SD-soft algorithm is trained with q values as follows:

$$q = p / m, \quad p = \{0.0, 0.2, 0.4, ..., 2.0\}, \qquad (20)$$

where p is the argument and m is the number of classes.

We train the SD-soft algorithm with 11 q values and the SD algorithm on 30 UCI data sets. The leaf classifiers used are the naive Bayes classifier, the C4.5 algorithm, and SVM. From Eq. (20), q value is adjusted by p value; therefore, we perform experiments with different p values. We first compare average test accuracies of the SD algorithm with those of the SD-soft algorithm with different p values. Average test accuracies are shown in Fig. 8. The best p value we select for a data set is the one that makes the SD-soft algorithm obtain the best average test accuracy on this data set. Then the SD-soft algorithm with the best q value is compared with the SD algorithm. The *t*-test results are listed in Table 6.

From Fig. 8, when using the naive Bayes classifier or SVM as the leaf classifier, the SD-soft algorithm obtains worse test accuracies than the SD algorithm. However, when using the C4.5 algorithm as the leaf classifier and when the p value is 0.6, 0.8, or 1.0, the SD-soft algorithm obtains better test accuracies.

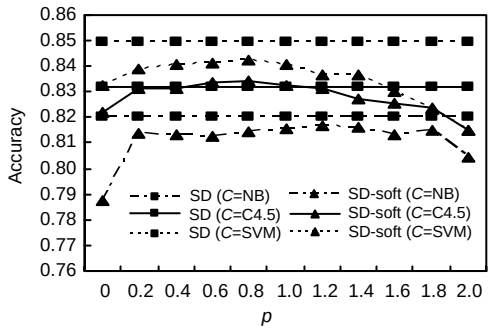

Fig. 8 Test accuracies of the SD and SD-soft algorithms with different *p* values

Table 6 shows that the SD-soft algorithm can obtain better generalization ability than the SD algorithm when the best average test accuracy is selected for each data set. Thus, it is possible for the SD-soft

algorithm to obtain better generalization ability than the SD algorithm if an appropriate q value is selected.

Table 6 The *t*-test results (W/L/T) of comparing the SD-soft algorithm with the SD algorithm

Algorithm	W/L/T
SD-soft (C=NB) vs. SD (C=NB)	12/1/17
SD-soft (C=C4.5) vs. SD (C=C4.5)	15/0/15
SD-soft (C=SVM) vs. SD(C=SVM)	10/4/16

5 Discussion

5.1 Other two SD-soft algorithms

The SD-soft algorithm is proposed to deal with the overlapped regions generated by the naive Bayes classifier, and adopts the method that the divide-and-conquer algorithm uses to deal with the overlapped regions. In this section, we investigate two other methods of dealing with the overlapped regions.

We first use Eq. (21) to obtain the data set $\{D_{si}\}$ of the SD-soft algorithm, and the resulting algorithm is named SD-soft.v2:

$$D_{si} = D_i \cup \{(\boldsymbol{x}, y) \mid \mathrm{NB}(\boldsymbol{x}) \neq Y_i, p_{\mathrm{NB}}(Y_i \mid \boldsymbol{x}) > q\}. \quad (21)$$

When $q \geq 0.5$, data set D_{si} is the same as data set D_i, and thus the SD-soft.v2 algorithm uses the prediction method of the SD algorithm to make the prediction.

The SD-soft and the SD-soft.v2 algorithms are both based on the conditional probability $p_{\mathrm{NB}}(Y_i|\boldsymbol{x})$. Next we propose another SD-soft algorithm based on the prediction rank of label Y_i.

The function $r(\mathrm{NB}, \boldsymbol{x}, Y_i)$ calculates the prediction rank of label Y_i:

$$r(\mathrm{NB}, \boldsymbol{x}, Y_i) = \left|\{Y_j \mid p_{\mathrm{NB}}(Y_j \mid \boldsymbol{x}) > p_{\mathrm{NB}}(Y_i \mid \boldsymbol{x})\}\right| + 1. \quad (22)$$

We use Eq. (23) to obtain the data set $\{D_{si}\}$ of the SD-soft algorithm, and the resulting algorithm is named SD-soft.v3:

$$D_{si} = \{(\boldsymbol{x}, y) \mid r(\mathrm{NB}, \boldsymbol{x}, Y_i) \leq k, p_{\mathrm{NB}}(Y_i \mid \boldsymbol{x}) > 0\}. \quad (23)$$

When k=1, data set D_{si} is the same as data set D_i, and thus the SD-soft.v3 algorithm uses the prediction method of the SD algorithm to make the prediction.

We compare the above two new SD-soft algorithms with the SD algorithm. The SD-soft.v2 algorithm uses the q value in Eq. (20). The k values of the SD-soft.v3 algorithm are 1, 2, and 3. For these two new algorithms, we select the best average test accuracy amongst different argument values for each data set, and compare it with the test accuracy of the SD algorithm (Table 7).

Table 7 The *t*-test results (W/L/T) of comparing two SD-soft algorithms with the SD algorithm

Algorithm	W/L/T
SD-soft.v2 (C=NB) vs. SD (C=NB)	10/1/19
SD-soft.v2 (C=C4.5) vs. SD (C=C4.5)	11/0/19
SD-soft.v2 (C=SVM) vs. SD (C=SVM)	7/2/21
SD-soft.v3 (C=NB) vs. SD (C=NB)	9/0/21
SD-soft.v3 (C=C4.5) vs. SD (C=C4.5)	6/0/24
SD-soft.v3 (C=SVM) vs. SD (C=SVM)	8/0/22

From Table 7, these two new algorithms can both obtain better generalization abilities than the SD algorithm. Comparing Table 7 with Table 6, we find that they cannot obtain better generalization abilities than the SD-soft algorithm, except that when the SD-soft.v3 algorithm adopts SVM as the leaf classifier. Like the SD-soft algorithm, argument selection is crucial for the above two new algorithms.

5.2 Applying the SD algorithm to the AODE algorithm, the C4.5 algorithm, and SVM

We first adopt the SD algorithm to train the C4.5 algorithm and SVM in decision regions of the AODE algorithm (Table 8).

According to Table 8, training C4.5/SVM in decision regions of the AODE algorithm by the SD algorithm can improve generalization abilities of both the AODE algorithm and leaf classifiers.

Table 8 The *t*-test results (W/L/T) of applying the SD algorithm to the AODE algorithm

Algorithm	W/L/T
SD (AODE+C4.5) vs. AODE	12/4/14
SD (AODE+C4.5) vs. C4.5	16/3/11
SD (AODE+SVM) vs. AODE	16/2/12
SD (AODE+SVM) vs. SVM	11/8/11

We then adopt the SD algorithm to train the C4.5 algorithm, the naive Bayes classifier, and SVM in decision regions of the C4.5 algorithm and SVM. The *t*-test results are listed in Tables 9 and 10, respectively.

Table 9 The *t*-test results (W/L/T) of applying the SD algorithm to the C4.5 algorithm

Algorithm	W/L/T
SD (C4.5+C4.5) vs. C4.5	6/1/23
SD (C4.5+NB) vs. C4.5	17/2/11
SD (C4.5+NB) vs. NB	12/14/4
SD (C4.5+SVM) vs. C4.5	15/4/11
SD (C4.5+SVM) vs. SVM	4/17/9

Table 10 The *t*-test results (W/L/T) of applying the SD algorithm to SVM

Algorithm	W/L/T
SD (SVM+SVM) vs. SVM	8/5/17
SD (SVM+C4.5) vs. SVM	5/6/19
SD (SVM+C4.5) vs. C4.5	17/6/7
SD (SVM+NB) vs. SVM	4/4/22
SD (SVM+NB) vs. NB	16/8/6

Tables 9 and 10 demonstrate that the SD algorithm is applicable for neither the C4.5 algorithm nor SVM. SD (C4.5+C4.5) improves the generalization ability of the C4.5 algorithm in only six data sets. SD (C4.5+NB) and SD (C4.5+SVM) both improve the generalization ability of the C4.5 algorithm, but they reduce the generalization abilities of the naive Bayes classifier and SVM. SD (SVM+SVM) improves the generalization ability of SVM in eight data sets, but reduces it in five data sets. SD (SVM+C4.5) and SD (SVM+NB) cannot improve the generalization ability of SVM, although they improve the generalization abilities of the C4.5 algorithm and the naive Bayes classifier in over half the data sets.

Thus, the SD algorithm is applicable for the AODE algorithm, but it is applicable for neither the C4.5 algorithm nor SVM.

5.3 Why does the SD algorithm work?

From experimental results of Sections 4 and 5.2, the SD algorithm is applicable for both the naive Bayes classifier and the AODE algorithm, but it is not applicable for either the C4.5 algorithm or SVM. In this section we attempt to investigate the cause.

Huang *et al*. (2008) divided classifiers into global learning and local learning. Global learning makes descriptions of data, whereas local learning captures the local useful information from data. Global learning aims to estimate a distribution of data, whereas local learning aims to obtain decision boundaries for classification. The disadvantage of global learning lies in that it may not make good use of the information contained in instances near decision boundaries. The disadvantage of local learning lies in that it may not make good use of the global information contained in the whole data set.

The naive Bayes classifier and the AODE algorithm belong to global learning, and the C4.5 algorithm and SVM belong to local learning. The naive Bayes classifier and the AODE algorithm make good use of the global information, but may not make good use of the local information near decision boundaries. The C4.5 algorithm and SVM make good use of the local information, but may not make good use of the global information. For the naive Bayes classifier and the AODE algorithm, training the C4.5 algorithm and SVM in their decision regions can make good use of the local information contained in instances near decision boundaries. For the C4.5 algorithm and SVM, being trained in decision regions of the naive Bayes classifier and the AODE algorithm can make good use of the global information contained in the whole data set. From Section 5.2, it cannot improve the generalization abilities of the C4.5 algorithm and SVM by training the naive Bayes classifier, the C4.5 algorithm, and SVM in decision regions of the C4.5 algorithm and SVM, which indicates that it cannot obtain better generalization abilities to train classifiers in decision regions of the local learning classifiers by applying the SD algorithm.

The NBTree algorithm can be regarded as training the naive Bayes classifier in decision regions of the C4.5 algorithm and obtains better generalization ability than both classifiers (Kohavi, 1996). However, it differs from using the SD algorithm to train the naive Bayes classifier in decision regions of the C4.5 algorithm (SD(C4.5+NB)). First, the NBTree algorithm trains the naive Bayes classifier in regions generated by the C4.5 algorithm, not the real decision regions of the C4.5 algorithm. Second, when training the C4.5 algorithm, the NBTree algorithm takes the test accuracy of the naive Bayes classifier into consideration, whereas the SD algorithm does not. Thus, the generalization ability of the NBTree algorithm is better than that of SD(C4.5+NB). The success of the NBTree algorithm indicates that it will need much more complex methods to train classifiers in decision regions of classifiers belonging to local learning.

6 Conclusions

This paper investigates methods of training classifiers in decision regions of the naive Bayes classifier. The SD and three SD-soft algorithms all generate two-level classifier trees with the naive Bayes classifiers as root nodes.

Experimental results on 30 UCI data sets demonstrate the effectiveness of these four new algorithms: SD, SD-soft, SD-soft.v2, and SD-soft.v3. When using the C4.5 algorithm and SVM as leaf classifiers, the SD algorithm can obtain better generalization ability than the naive Bayes classifier, leaf classifiers, the NBTree, and the AODE algorithms. When applied twice and using the naive Bayes classifier as leaf classifiers, the SD algorithm can obtain better generalization ability than the naive Bayes classifier and the NBTree algorithm. When using the C4.5 algorithm and SVM as leaf classifiers, the three SD-soft algorithms can obtain better generalization abilities than the SD algorithm, but argument selection is crucial for these three SD-soft algorithms.

The SD algorithm is also applicable for the AODE algorithm, but it is not applicable for the C4.5 algorithm or SVM.

The SD and SD-soft algorithms can make good use of the information contained in instances near decision boundaries, and the information may be ignored by global learning classifiers, such as the naive Bayes classifier and the AODE algorithm. The SD and SD-soft algorithms can be regarded as a new method of generating a hybrid of global learning and local learning.

Acknowledgements

We thank graduate students Zhen-zhen MI, Hai-tao WANG, Cheng-wei WANG, Qiang LIU, and Xiao-qiong WU of School of Computer Science and Technology at Zhejiang University for their advice in paper revision.

References

Bezdek, J.C., 1981. Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York, USA.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. Series: Information Science and Statistics. Springer-Verlag, New York, p.179-181.

Domingos, P., Pazzani, M., 1997. On the optimality of the simple Bayesian classifier under zero-one loss. *Mach. Learn.*, **29**(2-3):103-130. [doi:10.1023/A:1007413511361]

Frank, A., Asuncion, A., 2010. UCI Machine Learning Repository. School of Information and Computer Science, University of California, Irvine, CA, USA. Available from http://archive.ics.uci.edu/ml [Accessed on July 7, 2010].

Frank, E., Witten, I.H., 1998. Generating Accurate Rule Sets without Global Optimization. 15th Int. Conf. on Machine Learning, p.144-151.

Frosyniotis, D., Stafylopatis, A., Likas, A., 2003. A divide-and-conquer method for multi-net classifiers. *Pattern Anal. Appl.*, **6**(1):32-40. [doi:10.1007/s10044-002-0174-6]

Huang, K.Z., Yang, H.Q., King, I., Lyu, M., 2008. Machine Learning: Modeling Data Locally and Globally. Springer-Verlag, New York, p.1-28.

Kohavi, R., 1996. Scaling up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. 2nd Int. Conf. on Knowledge Discovery and Data Mining, p.202-207.

Mitchell, T.M., 1997. Machine Learning. WCB/McGraw-Hill, New York, p.14-15.

Pal, S.K., Mitra, S., 1992. Multi-layer perceptron, fuzzy sets, and classification. *IEEE Trans. Neur. Networks*, **3**(5):683-697. [doi:10.1109/72.159058]

Platt, J.C., 1999. Fast Training of Support Vector Machines Using Sequential Minimal Optimization. *In*: Scholkopf, B., Burges, C., Smola, A. (Eds.), Advances in Kernel Methods: Support Vector Machines. MIT Press, Cambridge, MA, USA, p.185-208.

Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA, USA.

Quinlan, J.R., 1996. Improved use of continuous attributes in C4.5. *J. Artif. Intell. Res.*, **4**(1):77-90.

Vapnik, V.N., 1995. The Nature of Statistical Learning Theory. Springer, Berlin Heidelberg.

Vlassis, N., Likas, A., 2002. A greedy EM algorithm for Gaussian mixture learning. *Neur. Process. Lett.*, **15**(1):77-87. [doi:10.1023/A:1013844811137]

Webb, G.I., Boughton, J.R., Wang, Z.H., 2005. Not so naive Bayes: aggregating one-dependence estimators. *Mach. Learn.*, **58**(1):5-24. [doi:10.1007/s10994-005-4258-6]

Witten, I.H., Frank, E., 2005. Data Mining: Practical Machine Learning Tools and Techniques (2nd Ed.). Morgan Kaufmann, San Francisco, CA, USA.

Wu, X.D., Kumar, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., *et al*., 2008. Top 10 algorithms in data mining. *Knowl. Inform. Syst.*, **14**(1):1-37. [doi:10.1007/s10115-007-0114-2]

Zheng, F., Webb, G.I., 2005. A Comparative Study of Semi-Naive Bayes Methods in Classification Learning. Fourth Australasian Data Mining Workshop, p.141-156.