Available online at www.sciencedirect.com
ScienceDirect
Computational Statistics & Data Analysis 52 (2008) 1928–1941
COMPUTATIONAL STATISTICS & DATA ANALYSIS
www.elsevier.com/locate/csda

A local boosting algorithm for solving classification problems

Chun-Xia Zhang*, Jiang-She Zhang

School of Science, Xi'an Jiaotong University, Xi'an Shaanxi 710049, People's Republic of China

Received 4 July 2006; received in revised form 9 May 2007; accepted 13 June 2007
Available online 29 June 2007

Abstract

Based on the *boosting-by-resampling* version of Adaboost, a local boosting algorithm for dealing with classification tasks is proposed in this paper. Its main idea is that in each iteration, a local error is calculated for every training instance and a function of this local error is utilized to update the probability that the instance is selected to be part of next classifier's training set. When classifying a novel instance, the similarity information between it and each training instance is taken into account. Meanwhile, a parameter is introduced into the process of updating the probabilities assigned to training instances so that the algorithm can be more accurate than Adaboost. The experimental results on synthetic and several benchmark real-world data sets available from the UCI repository show that the proposed method improves the prediction accuracy and the robustness to classification noise of Adaboost. Furthermore, the diversity-accuracy patterns of the ensemble classifiers are investigated by kappa-error diagrams.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Adaboost; Local boosting; Weak learning algorithm; Classification noise; Kappa-error diagram

1. Introduction

Prediction is one of the central supervised learning tasks in the fields such as data mining, machine learning and pattern recognition. It aims at predicting the most likely value $\hat{y}$ associated with a system under study, given the joint values of other variables $\mathbf{x} = (x_1, x_2, \ldots, x_p) \in \mathbf{R}^p$. The variables can be orderable (e.g., age, blood pressure, electric intensity) or categorical (e.g., sex, zip code, type of protocol for a network connection). When the response y is an orderable quantity (e.g., profit, income), the prediction problem is a *regression* task; when there is no natural order relation defined on the response space (e.g., y is a spam or non-spam email, fraudulent or normal network connection), one deals with a *classification* task.

With respect to the methods for solving classification problems, the techniques of combining multiple classifiers to produce a single classifier (see, for example, Wolpert, 1992; Latinne et al., 2002; Chandra and Yao, 2006; Banfield et al., 2007) have received much attention due to their potential to greatly increase prediction accuracy. The resulting composite classifier is generally referred to as an ensemble classifier and each individual classifier is often termed as a weak classifier or a base classifier. A great deal of literature on experimental and theoretical research (Hansen and Salamon, 1990; Optiz and Maclin, 1999; Tresp, 2001) has proved that the ensemble classifier is more accurate than any of its constituent member. Two popular methods for creating accurate ensemble classifiers are Bagging (Breiman, 1996) and Boosting (Schapire, 1990; Drucker and Cortes, 1996; Freund and Schapire, 1996, 1997; Breiman, 1998; Schapire

* Corresponding author. Tel.: +86 29 82673386.
E-mail addresses: cxzhang@mail.xjtu.edu.cn (C-X. Zhang), jszhang@mail.xjtu.edu.cn (J-S. Zhang).

0167-9473/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2007.06.015

and Singer, 1999; Friedman et al., 2000), both of which work by applying a weak learning algorithm[1] on different training sets that are obtained by permutating the original training set to construct their base classifiers. Furthermore, the efficiency of the ensemble classifier generation methods has been found to largely depend on the instability of the weak learning algorithm. Here an instable learning algorithm refers to that small permutations in its training data or in its construction can bring about large changes in the constructed predictor (Breiman, 1996, 1998). For example, classification tree and neural network are instable whereas K-nearest neighbor classifier and linear discriminant analysis are stable.

Bagging, an acronym of *bootstrap aggregating*, is a "bootstrap" ensemble method firstly proposed by Breiman (1996). It works by repeatedly training a given weak learning algorithm on different bootstrap samples (Efron and Tibshirani, 1993) where each bootstrap sample is generated by performing N extractions with replacement from the original training set of size N. As a result, in each of the resulting training sets for constructing weak classifiers, many of the original training instances may appear several times whereas others may never occur. The multiple classifiers are then combined using simple voting, that is, the final composite classifier predicts the class label of an instance $\mathbf{x}$ to the one most often assigned by the weak classifiers.

The resampling mechanism in Bagging leads to that the performance of each weak classifier is independent of one another and the weak classifiers can be built in parallel. However, the training set for each ensemble member of Boosting relies on the performance of the earlier trained classifiers. The probability of a specific instance being selected to be part of a particular classifier's training set depends on the performance of the previously built classifiers on that instance, that is, the probability distribution in each iteration is updated so that the instances being incorrectly predicted by previous classifiers can be chosen more often than those being correctly predicted. Thus, Boosting attempts to generate new classifiers that are able to better classify the "hard" instances for the previous ensemble members.

Since the advent of the first boosting algorithm, there have been various versions of boosting methods (see, for example, Schapire, 1990; Freund and Schapire, 1996, 1997; Breiman, 1998; Schapire and Singer, 1999; Friedman 2001, 2002) for solving classification problems. The original boosting approach is boosting by filtering developed by Schapire (1990). However, it requires a large amount of training data and is not feasible in many cases. This limitation is overcome by another boosting algorithm, Adaboost in several versions (Freund and Schapire, 1996, 1997), among which there are two familiar ones—*boosting-by-resampling* and *boosting-by-reweighting*. In *boosting-by-resampling*, the fixed training sample size and training instances resampled according to a probability distribution are used in each iteration; whereas in *boosting-by-reweighting*, all training instances with weights assigned to each instance are used in each iteration to train the base classifier and this technique is only useful when the weak learning algorithm can handle weighted instances.

However, there is increasing evidence to show that Adaboost is susceptible to noise although it has become a very popular classification method over the past years for its simplicity and adaptivity. One of the most convincing experimental studies that find this phenomenon is reported by Dietterich (2000). In his experiments, Dietterich compares the performance of Adaboost and Bagging on some benchmark classification data sets and investigates the dependence of the performance of these methods on the addition of classification noise to the training data. As expected, the prediction accuracy of both Adaboost and Bagging deteriorates as the amount of noise increases. However, the error rate of Adaboost increases more substantially than that of Bagging. Dietterich (2000) also gives an explanation of this phenomenon: Adaboost tends to assign much higher weight to noisy instances than other ones and this is more serious in its later iterations. As a result, hypotheses generated in later iterations overemphasize the noisy training instances and cause the performance of the combined hypothesis to deteriorate. Maclin and Optiz (1997) and Rätch et al. (2001) have also found the similar phenomenon for Adaboost. Therefore, some technique needs to be employed to enhance the robustness of Adaboost to classification noise.

Note that in the *boosting-by-resampling* version of Adaboost (Freund and Schapire, 1996), a global error is computed in each iteration as the sum of probabilities that the currently trained classifier incorrectly predicts the training instances with respect to the probability distribution over the training set. In each iteration, the probability distribution is updated by a function of the global error according to the performance of the trained classifier. Considering that whether a classifier correctly or incorrectly classifies a specific instance is closely related to its performance on the neighbors of

[1] In terms of PAC (probably approximately correct) learning theory (Kearns and Vazirani, 1994), a weak learning algorithm is defined as an algorithm that, given $\varepsilon \leqslant \frac{1}{2} - \gamma$ ($\gamma > 0$) and $\delta > 0$, can achieve an error rate ε which is at least slightly better than random guessing, with a probability at least $1 - \delta$.

that instance, a local boosting algorithm for solving classification problems is proposed in this paper. Its main idea is as follows: in each iteration, a local error is calculated for every training instance and a function of this local error is utilized to update the probability that the instance appears in the training set of next classifier. When classifying a novel instance, the similarity information between it and each training instance is taken into account. Meanwhile, a parameter is introduced into the process of updating the probabilities assigned to training instances so that the algorithm can be more accurate than Adaboost. The performance of the proposed method is examined and compared with that of several other classification methods on synthetic and some real-world data sets. Furthermore, the diversity-accuracy patterns of the ensemble classifiers are investigated by kappa-error diagrams.

The remainder of this paper is organized as follows. Section 2 describes the proposed local boosting algorithm for dealing with classification tasks in detail. In Section 3, the performance of the proposed algorithm is examined and compared with that of other classification methods by some simulations. Some benchmark real-world data sets from the UCI repository are analyzed by the proposed method as well as other classification methods in Section 4. Section 5 includes the conclusions of this paper.

2. Local boosting algorithm

Let $\mathscr{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ denote a training set of N independent instances consisting of input attributes $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip}) \in \mathbf{X} \subset \mathbb{R}^p$ and class labels $y_i \in Y = \{\omega_1, \omega_2, \ldots, \omega_J\}$. In a classification problem, the goal is to use the information only from $\mathscr{L}$ to construct classifiers that have good generalization ability, namely, perform well on the previously unseen data. Since multi-class classification problems can be generally transformed into many binary ones to solve (for example, see Freund and Schapire, 1997), we assume in this paper that the label space Y consists of just two possible labels, that is, $Y = \{-1, +1\}$. For simplicity of the notations, we denote the *boosting-by-resampling* version of Adaboost and the proposed method, respectively, by the global and local boosting methods in the following discussions. Before we embark on proposing the local boosting algorithm, we firstly list the pseudocode of the global boosting algorithm in Fig. 1 as follows.

In the global boosting algorithm, one of the main ideas of it is to maintain a probability distribution or a set of weights D_t over the training set $\mathscr{L}$. In each iteration, say t, the algorithm takes as input a training set $\mathscr{L}_t = \{(\mathbf{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^N$ which is randomly selected from $\mathscr{L}$ according to D_t and the weight of the ith training instance assigned by D_t is denoted by $D_t(i)$. Initially, all weights are set to be equal, but in each sequential iteration, the weights of incorrectly classified instances are increased in a suitable way so that the following trained classifier focuses on the "hard" instances in the original training set $\mathscr{L}$.

It should be noted that in the global boosting algorithm, the performance of a weak classifier h_t is evaluated by its misclassification error over all training instances with respect to the distribution D_t. That is to say, ε_t calculated in (1) is a global measure of the performance of h_t on $\mathscr{L}$. Considering that h_t performs well on some training instances whereas behaves badly on the others, however, a local measure to evaluate how well h_t performs should be established. Since the performance of h_t on a particular instance is generally closely related to its performance on the neighbors of that instance, here we propose a local boosting algorithm for solving classification problems through defining a neighborhood for each training instance. The detailed steps of this algorithm are given in Fig. 2.

In each iteration of the local boosting method, for every training instance i $(i = 1, 2, \ldots, N)$, the sum of probabilities that h_t misclassifies the neighbors of it is taken as a measure of the performance of h_t on the ith instance. If the input attributes $X_1, X_2, \ldots, X_p$ are all continuous, the neighborhood $N(i)$ used in the above algorithm can be defined as follows: for the ith instance, compute the Euclidian distances from all training instances to it and sort them, then select k closest ones to it, where k indicates the number of training instances contained in $N(i)$. If some attributes are categorical, the distance should be computed cautiously. With respect to a categorical attribute, we define the difference between two instances to be 0 if they belong to the same category and to be 1 otherwise. After quantifying all categorical attributes, the Euclidian distance is calculated and $N(i)$ is similarly selected. Considering that the efficiency of the proposed algorithm may be influenced by different values of k, we will conduct some simulations in Section 3 to investigate this problem.

Since an important issue that has been explored by many previous researchers is the question of how well the ensemble methods will perform in situations where there is some amount of classification noise (for example, see Maclin and Optiz, 1997; Dietterich, 2000) and they found that the global boosting algorithm seems to be susceptible to noise. They explained the reasons for this phenomenon to be that more and more weight is placed on the noisy instances to force

- **Given**: a training set $\mathscr{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ in which $\mathbf{x}_i \in \mathbf{X} \subset \mathbb{R}^p$ and $y_i \in Y = \{-1, +1\}$; a weak learning algorithm $\mathscr{W}$; number of iterations T.
- **Initialize**: set the probability distribution over $\mathscr{L}$ as $D_1(i) = 1/N$ $(i = 1, 2, \cdots, N)$.
- **Iterate**:

  For $t = 1, \cdots, T$:

  1. According to the distribution D_t, draw N training instances at random from $\mathscr{L}$ with replacement to compose a new training set $\mathscr{L}_t = \{(\mathbf{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^N$.
  2. Apply $\mathscr{W}$ to $\mathscr{L}_t$ to train a weak classifier $h_t : \mathbf{X} \to \{-1, +1\}$ and compute the error of h_t as
  $$\varepsilon_t = \Pr_{i \sim D_t}(h_t(\mathbf{x}_i) \neq y_i) = \sum_{i=1}^{N} I(h_t(\mathbf{x}_i) \neq y_i) D_t(i), \tag{1}$$
  where $I(\cdot)$ is the indicator function which takes value 1 or 0 depending on whether the ith training instance is misclassified by h_t or not.
  3. If $\varepsilon_t > 0.5$, then set $D_{t+1}(i) = 1/N$ $(i = 1, 2, \cdots, N)$ and goto step 1; if $\varepsilon_t = 0$, then set $\varepsilon_t = 10^{-10}$ to continue the following iterations.
  4. Choose $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
  5. Update the probability distribution over $\mathscr{L}$ as
  $$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t}, & \text{if } h_t(\mathbf{x}_i) = y_i \\ e^{\alpha_t}, & \text{if } h_t(\mathbf{x}_i) \neq y_i \end{cases} \\ &= \frac{D_t(i)\exp(-\alpha_t y_i h_t(\mathbf{x}_i))}{Z_t}, \end{aligned} \tag{2}$$
  where Z_t is a normalization factor (it should be chosen so that D_{t+1} is a distribution over $\mathscr{L}$).

  Endfor

- **Output**: the ensemble classifier as
$$H(\mathbf{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right). \tag{3}$$

Fig. 1. Global boosting algorithm.

the weak classifier to agree with them; thus, the weak learning algorithm obtains erroneous hypotheses that degrade the generalization ability of the ensemble classifier. Therefore, a parameter β is introduced into the process of updating the probability distribution to alleviate this problem to some extent. In fact, the role of parameter β playing in the local boosting procedure is to make the decrement (increment) of the probabilities assigned to the correctly (incorrectly) predicted instances to be smaller than that in the global boosting method. Note that the larger β, the more similar the probabilities of all training instances being updated regardless of whether they are correctly or incorrectly predicted. The smaller β, the more different between the decrement of the probabilities for correctly classified instances and the increment of the probabilities for misclassified ones. The experiments we have done demonstrate that too large

- **Given**: a training data set $\mathscr{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbf{X} \subset \mathbb{R}^p$ and $y_i \in Y = \{-1, +1\}$; a weak learning algorithm $\mathscr{W}$; number of iterations T; positive constant β; size of neighborhood $N(\cdot)$.
- **Initialize**: set the probability distribution over $\mathscr{L}$ as $D_1(i) = 1/N$ $(i = 1, 2, \cdots, N)$.
- **Iterate**:

  For $t = 1, \cdots, T$:

  1. According to the distribution D_t, extract N training instances at random from $\mathscr{L}$ with replacement to compose a new training set $\mathscr{L}_t = \{(\mathbf{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^N$.
  2. Train $\mathscr{L}_t$ to the given weak learning algorithm $\mathscr{W}$ to get classifier $h_t : \mathbf{X} \to \{-1, +1\}$ and compute the local error of h_t at the ith training instance as
  $$\varepsilon_t(i) = \sum_{j \in N(i)} I(h_t(\mathbf{x}_j) \neq y_j) D_t'(j), \quad \text{where} \quad D_t'(j) = \frac{D_t(j)}{\sum_{l \in N(i)} D_t(l)}, \tag{4}$$
  in which $N(i)$ indicates the neighborhood of the ith training instance and $I(\cdot)$ is the indicator function.
  3. If $\varepsilon_t(i) > 0.5$, then set $D_{t+1}(i) = 1/N$ $(i = 1, 2, \cdots, N)$ and goto step 1; if $\varepsilon_t(i) = 0$, then set $\varepsilon_t(i) = 10^{-10}$ to continue the following iterations.
  4. Let $\alpha_t(i) = \frac{1}{2} \times \ln\left(\frac{1-\varepsilon_t(i)}{\varepsilon_t(i)}\right)$.
  5. Update the probability distribution over $\mathscr{L}$ as
  $$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t(i)/\beta}, & \text{if } h_t(\mathbf{x}_i) = y_i \\ e^{\alpha_t(i)/\beta}, & \text{if } h_t(\mathbf{x}_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-(\frac{\alpha_t(i)}{\beta}) y_i h_t(\mathbf{x}_i))}{Z_t}, \end{aligned} \tag{5}$$
  where Z_t is a normalization factor (it should be chosen so that D_{t+1} is a distribution on $\mathscr{L}$).

  Endfor

- **Output**: the ensemble classifier as
$$H(\mathbf{x}) = \operatorname{sign}\left(\sum_{t=1}^{T}\left(\sum_{i=1}^{N} \frac{\alpha_t(i)}{d^2(\mathbf{x}, \mathbf{x}_i)}\right) h_t(\mathbf{x})\right), \tag{6}$$
where $d^2(\mathbf{x}, \mathbf{x}_i)$ denotes the distance between the new instance to be predicted and the ith training instance.

Fig. 2. Local boosting algorithm.

β is not beneficial to the local boosting algorithm. In practice, the value of β should be cautiously selected and it can be determined by some data-driven methods such as a validation set according to the specific classification tasks at hand.

Once a series of weak classifiers h_t $(t = 1, 2, \ldots, T)$ has been trained, each of which has a weight coefficient $\alpha_t(i)$ at every training instance i $(i = 1, 2, \ldots, N)$. Therefore, when combining multiple classifiers in some way to predict a new instance, we take the sum of each instance's weight coefficient divided by the squared distance from the new instance to each training instance, that is, $\sum_{i=1}^{N} \alpha_t(i)/d^2(\mathbf{x}, \mathbf{x}_i)$, as a measure of the importance of $h_t(\mathbf{x})$ in the ensemble classifier. Compared with the global boosting algorithm, the proposed method utilizes more information of training data so the prediction accuracy can be expected to be improved.

As a matter of fact, $1/d^2(\mathbf{x}, \mathbf{x}_i)$ can be seen as a measure of similarity between the new instance and the ith training instance. Put in another way, larger $1/d^2(\mathbf{x}, \mathbf{x}_i)$ indicates that the ith training instance is more influential to the new instance. In this case, the coefficient $\alpha_t(i)$ of h_t should be weighted in some way to reflect this information and the importance of h_t in the final composite classifier should be the sum of the weighted values of $\alpha_t(i)$ over all training instances. Recall that we just take the measure of similarity $1/d^2(\mathbf{x}, \mathbf{x}_i)$ as a weight assigned to $\alpha_t(i)$, other complex similarity measures such as the Gaussian function of $d^2(\mathbf{x}, \mathbf{x}_i)$ can also be used to achieve the same task. Here we omitted the details about different similarity measures to avoid distracting from the central theme of this paper.

3. Simulation studies

In this section, we conduct some simulations to examine the performance of the proposed local boosting method. There are mainly three purposes for conducting these simulations. Firstly, the influence of the parameter β and the number of instances contained in $N(i)$ to the prediction accuracy of the local boosting algorithm should be investigated. Secondly, the performance of the proposed method will be examined and compared with that of several other classification methods. And thirdly, we want to know how well the considered classification methods perform in situations where there is some amount of classification noise.

Because the random function generator introduced by Friedman (2001, 2002) creates prediction targets with various levels of difficulties and spans a broad range of problems, we take this appropriate choice for our aforementioned task. The following used data consist of 1750 observations $(\mathbf{x}_i, y_i)$ with

$$y_i = \operatorname{sign}(F^*(\mathbf{x}_i) - \operatorname{median}_{\mathbf{x}} F^*(\mathbf{x})) \quad \text{and} \quad F^*(\mathbf{x}_i) = \sum_{l=1}^{20} a_l g_l(\mathbf{z}_l), \tag{7}$$

in which the number of input variables was taken to be $n = 10$ and their joint distribution was taken to be standard normal, i.e., $\mathbf{x} \sim N(\mathbf{0}, \mathbf{I})$. The coefficient $\{a_l\}_{l=1}^{20}$ in (7) were randomly generated from a uniform distribution on interval $(-1, 1)$. Each $g_l(\mathbf{z}_l)$ is a function of a randomly selected subset, of size n_l, from the n-dimensional input variable vector $\mathbf{x}$ and n_l is a random number determined in another way. The detailed method to generate the data can refer to Friedman (2001). In all the simulations conducted below, the data were randomly split into a set $\mathscr{L}$ consisting of 750 observations for training and another set $\mathscr{T}$ with 1000 observations for testing. A classification tree (Breiman et al., 1984) was always adopted as the weak learning algorithm, in which the tree was iteratively built using the training set until one of the following terms happens: (1) there are less than 30 instances in a node; (2) all instances in a node belong to one class. The performance of one classification method was evaluated in terms of the misclassification error of the obtained classifier on the test set $\mathscr{T}$

$$\hat{e}(H) = \underset{(\mathbf{x}, y) \in \mathscr{T}}{E} I(H(\mathbf{x}) \neq y), \tag{8}$$

where $I(\cdot)$ is the indicator function. The following experiments were all conducted using the *Stats* package in Matlab software with version 7.1.

3.1. Effect of number of instances contained in $N(i)$

In this subsection, we shall examine the effect of number of training instances, say k, contained in $N(i)$ to the performance of the local boosting procedure. We took the value of k to be 20 with increment 20 to 380 and the value of β to be 5, 10, 15 and 20, respectively, to achieve our task. For each combination of the values for k and β, we conducted 10 trials through randomly dividing the training set $\mathscr{L}$ and testing set $\mathscr{T}$ and computed the misclassification error for each trial, then take the mean of them as a measure of the performance of the local boosting algorithm. The number of

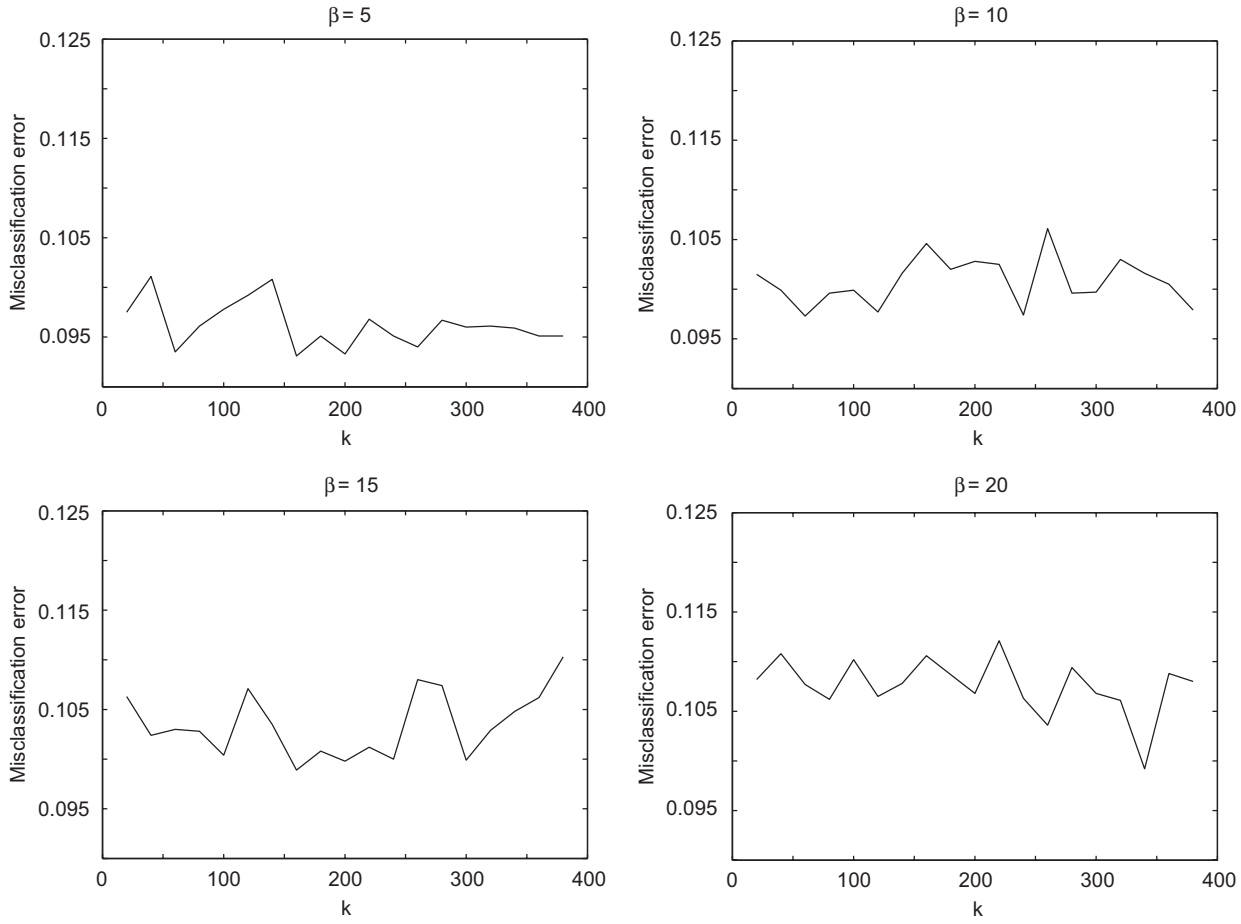


Fig. 3. Performance of the local boosting algorithm versus the number of instances contained in $N(i)$ for various values of β.

iterations was set to be 50 in each trial. Fig. 3 depicts how the computed mean misclassification error changes with the value of k for various values of β. In order to facilitate the comparisons, the scales labeled on the vertical axis of each subplot have been adjusted to be identical.

It is observed in Fig. 3 that for each fixed β, the performance of the proposed algorithm varies slightly with the value of k, which means that the effect of the choice of k is not very large. Therefore, the value of k can be taken to be small if we are interested in reducing the computational cost at the expense of a limited loss of prediction accuracy. Furthermore, it can also be found that the misclassification error is shifted upward somewhat when the value of β increases through comparing the plots obtained with different values of β.

3.2. *Influence of parameter β*

In order to evaluate the influence of the parameter β to the performance of the local boosting algorithm, we took the value of β to be 2.5 with increment 2.5 to 50, the number of instances contained in $N(i)$ to be 65, 125, 185, 245, respectively, and the number of iterations $T = 50$ to run some experiments similar to those done in Section 3.1. Fig. 4 shows the computed mean misclassification error as a function of parameter β for each considered value of k.

As seen in Fig. 4, the performance of the algorithm tends to deteriorate when the value of β increases and this phenomenon is similar for the cases where k takes different values. This finding is consistent with that in Fig. 3, that is, large value of β is not beneficial to the local boosting algorithm and it should be chosen to be relatively small.

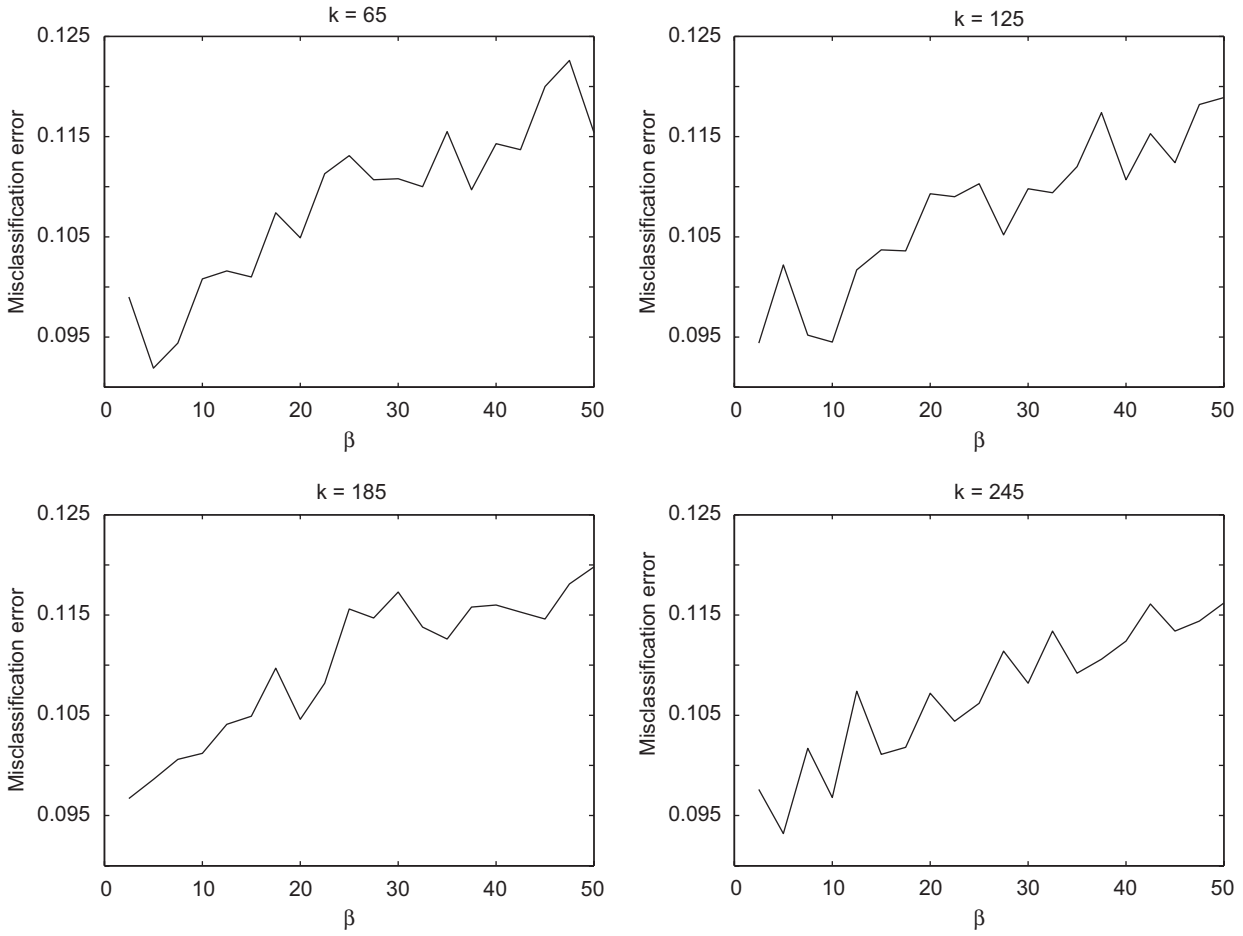


Fig. 4. Performance of the local boosting algorithm versus β for different numbers of instances contained in $N(i)$.

3.3. Comparisons with other classification methods

In order to comprehensively investigate the performance of the proposed algorithm, some simulations are further conducted in this subsection through comparing it with several other classification methods—single classification tree (Breiman et al., 1984), Bagging (Breiman, 1996), Random forest (Breiman, 2001) and Adaboost (Freund and Schapire, 1996). In order to facilitate the following discussions, we denote these considered methods and our proposed one respectively by SingleTree, Bagging, RandomForest, GlobalBoost and LocalBoost. For each algorithm, we conducted 100 trials through randomly dividing the training and testing sets and compute the misclassification error for each trial. For each of the ensemble classifier generation methods, 50 trees were combined in each trial to constitute its ensemble classifier. With respect to RandomForest, the parameter included in it was taken to be 2. The values of β and k used in LocalBoost were respectively taken to be 5 and 185. The resulting distributions of the misclassification error over 100 trials for each method are summarized by boxplot and they are depicted in Fig. 5.

The area in the rectangle of each boxplot shows the interquartile range and the midline indicates the median of the corresponding distribution. The upper (lower) bars outside of the rectangle represent the points closest to plus (minus) 1.5 interquartile range units from the upper (lower) quartiles. The isolated plus signs outside of each rectangle denote the outliers of the corresponding distribution. We can see from Fig. 5 that LocalBoost generally outperforms the other methods. Furthermore, it has the smallest error for 81 of 100 trials and on average its error rate was 0.86% higher than the best for each trial whereas GlobalBoost, RandomForest and Bagging were, respectively, the best on 13, 2 and 4 of the targets and their error rates were 15.45%, 32.88% and 25.48% larger than the best on average. SingleTree was never the best performer and on average it was 55.48% worse than the best.

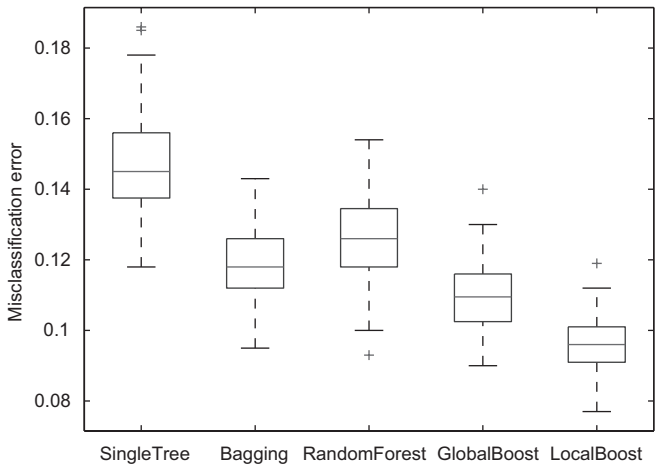


Fig. 5. Boxplots of the distributions of misclassification error for SingleTree, Bagging, RandomForest, GlobalBoost and LocalBoost.

Table 1
Mean and standard deviation of test errors for various methods on the simulated data with different levels of noise

Noisy level (%)	LocalBoost	SingleTree		Bagging		RandomForest		GlobalBoost	
0	9.70 ± 1.02	13.60 ± 0.89	•	11.33 ± 0.68	•	13.11 ± 0.83	•	10.93 ± 0.90	•
5	10.96 ± 1.02	16.79 ± 1.87	•	12.18 ± 0.84	•	12.77 ± 1.09	•	13.04 ± 0.92	•
10	12.68 ± 0.98	20.23 ± 2.50	•	13.92 ± 1.50	•	14.56 ± 1.34	•	17.02 ± 1.67	•
15	15.73 ± 1.10	24.29 ± 2.11	•	15.74 ± 1.03		15.62 ± 1.88		20.05 ± 1.02	•
20	17.81 ± 1.55	25.44 ± 2.34	•	17.37 ± 1.43		17.82 ± 1.75		24.35 ± 2.25	•

"•" indicates that LocalBoost is significantly better when significance level $\alpha = 0.05$.

3.4. Robustness to classification noise

Since robustness to classification noise is an important issue for a classification method, some simulations are conducted in this subsection to examine the robustness of the proposed boosting algorithm to noise and the obtained results are also compared with those of the other classification methods.

Here the used simulated data were identical to those utilized in the previous subsections. In order to add the classification noise at a given rate r, we randomly selected a fraction r of the data points without replacement from the original data and changed their class labels to be incorrect (the label for each instance was chosen uniformly at random from the incorrect labels). The noisy level r was taken, respectively, to be 0%, 5%, 10%, 15% and 20% to extensively explore the robustness of the local boosting as well as other classification methods to noise. At each level, the noisy simulated data were randomly split into a set with 750 observations for training and a set with 1000 observations for testing. This process was repeated 10 times, and then the average and standard deviation of the test errors were computed. In the experiments conducted here, the parameters utilized in every method were similar to those used in the previous subsections and Table 1 summarizes the experimental results.

In order to see whether LocalBoost is significantly better or worse than other methods, a one-tailed paired t-test was performed with significance level $\alpha = 0.05$ and the results for which a significant difference with LocalBoost was found are marked with a bullet or an open circle next to them. A bullet next to a result indicates that LocalBoost is significantly better than the respective method at the corresponding noisy level. An open circle next to a result denotes that LocalBoost performs significantly worse than the corresponding method.

In Table 1, LocalBoost is seen to be the best performer in most cases. When the noisy level is small, it performs significantly better than all the other methods. Although it behaves slightly worse than Bagging and RandomForest when the noise level is relatively large, the difference between them is not significant. These results show that in practice LocalBoost can be a preferential choice if no information about the level of classification noise is available.

3.5. Kappa-error diagrams

According to the simulation results obtained in the previous two subsections, LocalBoost is observed to perform satisfactorily, but the reasons for the its good performance is still not very clear and this problem will be investigated subsequently through analyzing the diversity-accuracy patterns of the generated ensemble classifiers.

As pointed out by many researchers (Hansen and Salamon, 1990; Optiz and Shavlik, 1996; Rodríguez et al., 2006), for an ensemble classifier to achieve better generalization ability than its each constituent member, it is critical that the ensemble classifier consists of highly accurate members which at the same time disagree as much as possible. That is to say, if an ensemble classifier has good performance, there should be a trade-off between diversity and accuracy of its ensemble members. Margineantu and Dietterich (1997) and Dietterich (2000) suggested a visualization means, kappa-error diagram, to investigate the diversity-accuracy pattern of the ensemble classifiers. For each pair of classifiers C_i and C_j in the ensemble, the accuracy of them is measured by the average of their errors estimated on the testing data set; the diversity between them is measured by the statistic kappa(κ) (Fleiss et al., 1981) which evaluates the level of agreement between two classifier outputs while correcting for chance. A kappa-error diagram is a scatter plot in which each point corresponds to a pair of classifiers C_i and C_j. On the x-axis of the plot is the diversity value κ and on the y-axis of it is the mean error of C_i and C_j, say, $E_{i,j} = (E_i + E_j)/2$.

The statistic κ is defined as follows. Suppose that there are J classes $\{\omega_1, \omega_2, \ldots, \omega_J\}$, κ is defined on the $J \times J$ coincidence matrix $\mathscr{M}$ of two classifiers C_i and C_j $(i, j = 1, 2, \ldots, T)$, where T is the number of classifiers in an ensemble. The entry $m_{l,s}$ of $\mathscr{M}$ is the proportion of the testing data set, in which classifier C_i labels as ω_l and classifier C_j labels as ω_s. Then the agreement between C_i and C_j can be measured as

$$\kappa_{i,j} = \frac{\sum_{l=1}^{J} m_{ll} - ABC}{1 - ABC}, \tag{9}$$

where $\sum_{l=1}^{J} m_{ll}$ is the observed agreement between the two classifiers C_i and C_j. "ABC", the acronym of "agreement-by-chance", is defined as

$$ABC = \sum_{l=1}^{J} \left(\sum_{s=1}^{J} m_{l,s} \right) \left(\sum_{s=1}^{J} m_{s,l} \right).$$

According to the above definition, $\kappa_{i,j} = 0$ when the agreement of C_i and C_j equals that expected by chance, and $\kappa_{i,j} = 1$ when C_i and C_j agree on every instance. Negative values occur when agreement is weaker than expected by chance. Since low values of $\kappa_{i,j}$ indicate higher diversity and small values of $E_{i,j}$ show better accuracy, the most desirable pairs of classifiers should lie in the bottom left corner of the scatter plot.

Fig. 6 shows the kappa-error diagrams of the ensemble classifiers constructed by the considered ensemble classifier generation methods at different noisy levels. With respect to each noisy level, we generated a training set with 750 observations to build the ensemble classifier and a testing set with 1000 observations to calculate the kappa-error diagram. All ensemble classifiers consist of 50 trees, therefore there are 1225 (C_{50}^2) points in each plot. In Fig. 6, the four plots displayed on the right of each noisy level correspond to the kappa-error diagrams of the ensemble classifiers constructed, respectively, by Bagging, RandomForest, GlobalBoost and LocalBoost. The number shown above each plot is the ensemble error estimated on the testing set. In order to facilitate the comparisons, the scales of κ and $E_{i,j}$ in the listed plots have been adjusted to be identical.

For each noisy level, we can see from Fig. 6 that Bagging gives a very compact cloud of points. Each point has a low error rate and a high value for κ, which indicates that the classifiers are accurate but not very diverse. RandomForest is more diverse than Bagging but is not as accurate as Bagging. GlobalBoost spans large diversity range but many classifier pairs have large errors. LocalBoost has diversity approximately equal to that of GlobalBoost, but better accuracy. These observations clearly show the trade-off between diversity and accuracy: as the classifiers become more diverse, they must become less accurate; conversely as they become more accurate, they must become less diverse. When more amount of classification noise is added, the errors for these methods all increase somewhat and the diversity of GlobalBoost and LocalBoost is hardly changed at all whereas that of Bagging and RandomForest is greatly increased resulting from the added noise. Based on this fact, one of the reasons for the satisfactory performance of LocalBoost may be that it achieves a better trade-off between diversity and accuracy of its ensemble members.

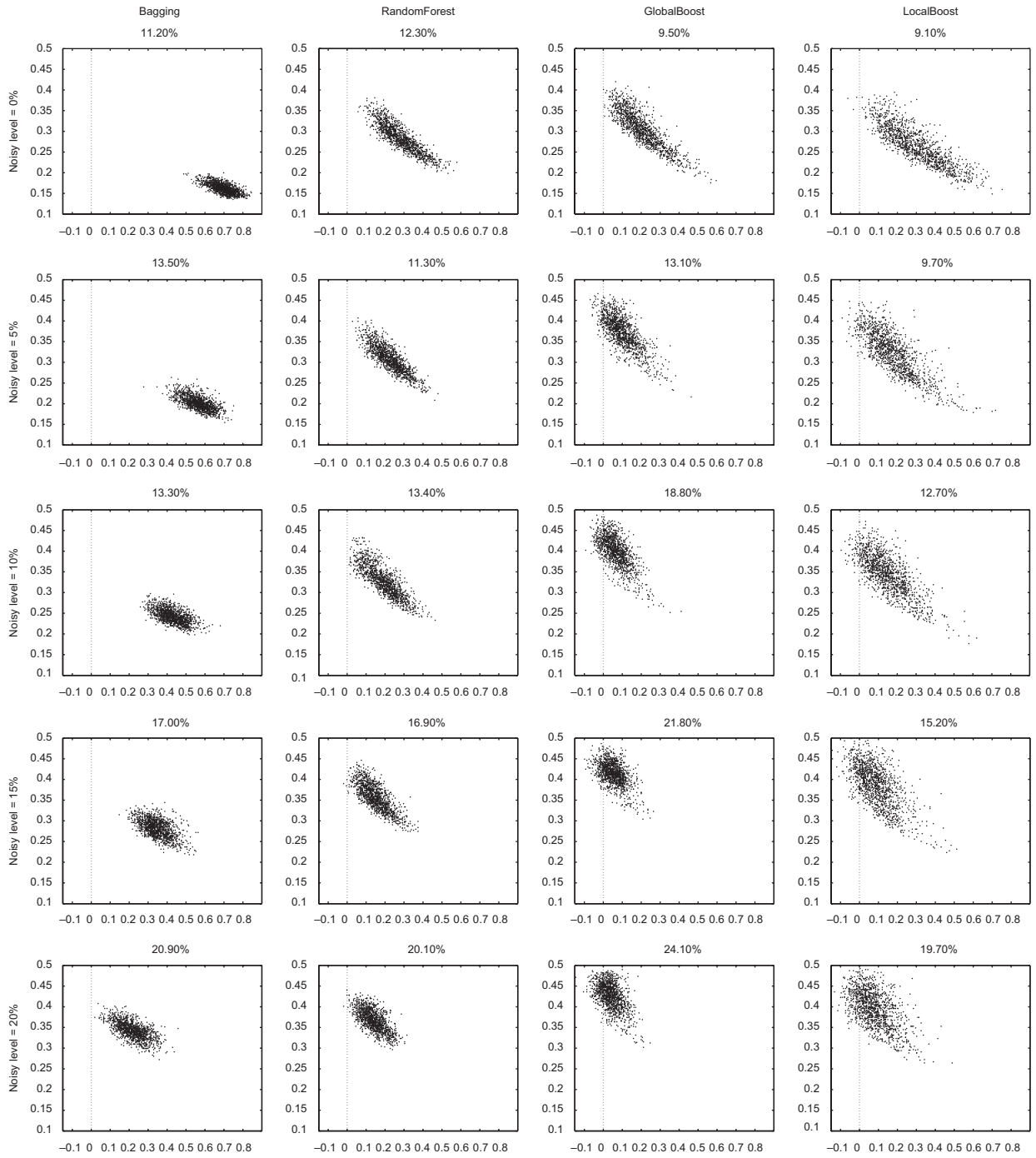


Fig. 6. Kappa-error diagrams of the ensemble classifiers constructed by Bagging, RandomForest, GlobalBoost and LocalBoost at different noisy levels.

4. Real-world data sets

In this section we examine the performance of LocalBoost and compare it with several other classification methods on some benchmark data sets available from the UCI repository (Blake and Merz, 1998). Some characteristics of the

Table 2
Summary of the used benchmark data sets

Data set	# Training	# Validation	# Testing	# Classes	# Input attributes		Missing values
					Discrete	Continuous	
Iris	80	20	50	3	0	4	–
Ionosphere	201	50	100	2	0	34	–
Hepatitis	105	20	30	2	13	6	×
Glass	124	40	50	7	0	10	–

Table 3
Mean and standard deviation of test errors for various methods on benchmark real-world data sets with different levels of classification noise

Noisy level (%)	Data set	LocalBoost	SingleTree		Bagging		RandomForest		GlobalBoost	
0	Iris	3.80 ± 1.99	6.20 ± 1.48	•	4.60 ± 2.84		5.00 ± 1.94		4.80 ± 2.35	
	Ionosphere	5.60 ± 1.26	13.00 ± 2.45	•	9.60 ± 3.24	•	8.30 ± 1.95	•	6.00 ± 2.49	
	Hepatitis	14.67 ± 6.32	16.67 ± 8.61		17.33 ± 6.05		15.67 ± 5.68		21.67 ± 7.07	•
	Glass	16.80 ± 4.24	23.60 ± 4.79	•	21.40 ± 6.67	•	22.80 ± 5.43	•	23.40 ± 4.62	•
5	Iris	6.40 ± 2.07	9.40 ± 6.47		6.80 ± 2.86		6.40 ± 2.80		9.00 ± 2.16	•
	Ionosphere	8.20 ± 4.02	15.40 ± 2.91	•	10.00 ± 2.98		9.60 ± 2.80		8.80 ± 1.69	
	Hepatitis	16.67 ± 8.31	16.33 ± 6.75		18.00 ± 5.71		19.67 ± 7.61		22.00 ± 10.09	
	Glass	11.20 ± 9.85	15.20 ± 10.25		13.40 ± 11.96		13.40 ± 7.18		17.80 ± 13.28	
10	Iris	8.60 ± 4.12	16.00 ± 5.16	•	10.00 ± 3.27		9.40 ± 6.11		12.40 ± 3.37	•
	Ionosphere	10.30 ± 3.30	18.70 ± 4.32	•	10.80 ± 3.43		9.80 ± 2.35		11.00 ± 3.09	
	Hepatitis	17.33 ± 7.98	19.00 ± 7.21		19.00 ± 6.49		20.67 ± 9.00		20.33 ± 7.77	
	Glass	13.20 ± 9.05	14.60 ± 4.22		13.20 ± 8.07		11.20 ± 8.70		14.20 ± 6.43	
15	Iris	13.60 ± 5.48	19.00 ± 5.83	•	13.40 ± 3.27		11.80 ± 2.57		17.60 ± 3.10	•
	Ionosphere	13.70 ± 3.86	19.70 ± 5.87	•	11.90 ± 3.11		11.40 ± 2.50		15.20 ± 5.31	
	Hepatitis	21.67 ± 6.71	24.67 ± 7.06		23.67 ± 8.23		18.33 ± 7.58		28.33 ± 10.21	
	Glass	8.40 ± 5.64	13.20 ± 6.55	•	17.00 ± 6.55	•	17.40 ± 7.49	•	11.00 ± 4.74	
	Iris	17.80 ± 7.86	22.60 ± 6.74		18.80 ± 5.75		17.20 ± 8.07		22.60 ± 7.89	
	Ionosphere	17.80 ± 5.41	25.60 ± 4.06	•	13.80 ± 5.94		10.04 ± 3.31	○	19.10 ± 3.18	
20	Hepatitis	25.00 ± 7.58	27.00 ± 10.24		18.00 ± 7.57	○	19.67 ± 9.49		37.33 ± 8.43	•
	Glass	9.60 ± 3.75	17.00 ± 4.24	•	10.40 ± 4.50		17.00 ± 8.12	•	15.00 ± 5.19	•

"•" indicates that LocalBoost is significantly better and "○" denotes that LocalBoost is significantly worse when significance level $\alpha = 0.05$.

considered data sets are described in Table 2. The column labeled "Missing values" denotes whether there are missing values in the corresponding data set, when this is true it is denoted by "×" and otherwise it is denoted by "–". For each data set, the three columns next to the name of it respectively gives the size of set used for training, validation and testing.

For each of the above data sets, we used the technique described in Section 3.4 to add some classification noise into it to investigate the robustness of the classification methods to noise. When adding the noise of level r, a fraction r of the data points was randomly selected from the original data without replacement and their class labels were changed to be incorrect. The value of r was taken to be 0%, 5%, 10%, 15%, and 20%, respectively, to explore the resistance of the classification procedures to noise. For each of the obtained noisy data set, we randomly split it into three sets for training, validation and testing and this process was repeated 10 times with different random number generating seeds in order to obtain consistent results. In all experiments, 50 classification trees were utilized to construct the ensemble classifier by each considered classifier combination technique. It is worthwhile to point out that the validation set was used to determine the optimal values of parameters β and k included in LocalBoost and for the other classification methods the classifier was trained on the combination of training and validation sets.

We extended our binary-class version of LocalBoost to solve multi-class problems using the idea similar to Adaboost.M1 (Freund and Schapire, 1997). The experimental results were reported in Table 3. The first two columns

show the noisy level and the names of data sets and the other columns denote the mean and standard deviation of test errors estimated by the classification methods labeled in the first row. In order to see whether LocalBoost is significantly better or worse than other methods, a one-tailed paired t-test was performed with significance level $\alpha = 0.05$ and the results for which a significant difference with LocalBoost was found are marked with a bullet or an open circle next to them.

We can see from Table 3 that, LocalBoost is generally the superior performer. It always performs better than SingleTree and GlobalBoost in all the considered cases and sometimes the difference between them is significant. When the amount of noise is large, LocalBoost is seen to perform sometimes worse than Bagging and RandomForest and the statistically significant differences are unfavorable for it in two cases. However, LocalBoost behaves better than them when noisy level is small and the significant differences are favorable for it in six cases.

On the first three data sets *iris*, *ionosphere* and *hepatitis*, the performance of each procedure deteriorates to some degree as the amount of classification noise increases especially when the noisy level r is large. However, the prediction accuracy of LocalBoost decreases more slowly than that of GlobalBoost, which is seen to be even worse than SingleTree in some situations. This demonstrates that the performance of GlobalBoost is susceptible to noise and it is consistent with the results obtained by previous researchers (Maclin and Optiz, 1997; Dietterich, 2000).

Furthermore, we find a surprising phenomenon on the *glass* data: the error rates of all methods *decline* somewhat with the increase of classification noise, which is different from the other three real-world data sets. This may be the reason that this classification problem belongs to a multi-class one which contains seven classes and there are relatively few even no instances for some class labels, for example, there are no instances for class label "4" in the original data. When the data set is not noisy the class boundaries are complex, which results in large error rates for all considered classification methods and as can be seen from the fifth row of Table 3. With classification noise added into it, the class boundaries become relatively simple and this leads to the improvement of the error rates.

5. Conclusions

In this paper, we proposed a local boosting algorithm based on the *boosting-by-resampling* version of Adaboost considering that whether a classifier correctly or incorrectly predicts a specific instance is closely related to its performance on the neighbors of that instance. A parameter is introduced into the process of updating the probabilities assigned to training instances in each iteration so that the algorithm is more accurate than Adaboost. When classifying a novel instance, the similarity information between it and each training instance is also taken into account. The experimental results on synthetic and several benchmark real-world data sets from the UCI repository demonstrate that the local boosting algorithm performs better in most of the considered cases. Furthermore, the results on data sets with different levels of artificial classification noise show that the local boosting procedure is more robust to noise than Adaboost. One of the reasons for the satisfactory performance of the local boosting algorithm found by analyzing the kappa-error diagrams of the ensemble classifiers is that the local boosting algorithm achieves a better trade-off between diversity and accuracy of its ensemble members. Although this paper is only an empirical study, the proposed method may be of some practical uses for solving classification problems with the support of the simulation results.

It should be recalled that when employing the proposed local boosting algorithm to solve a practical problem, the value of parameter β should be selected to be not too large and that of k, namely, number of instances contained in the neighborhood of a training instance, can be chosen to be a small or moderate value since the influence of it to the performance of the local boosting method is not very significant. In order to obtain the optimal classification accuracy, the values of these parameters can be selected via a validation set.

In fact, the techniques used in this paper to modify the *boosting-by-resampling* version of Adaboost can also be applied to the *boosting-by-reweighting* version of Adaboost and how the obtained method will perform is deserved to be investigated. Furthermore, the weak learning algorithm we utilized in this paper is a classification tree, how the local boosting procedure will behave when using other choices of weak learning algorithm such as neural network is one of our future tasks. The influence of different measures of similarity between training instances and that of other ways to define the neighborhood $N(i)$ to the performance of the local boosting algorithm are also some aspects deserved to be studied further.

Acknowledgments

The authors would like to express their gratitude to the associate editor and one anonymous referee for their valuable comments and suggestions which lead to a substantial improvement of this paper. This work was supported in part by the National Natural Science Foundations of China (No. 10531030 and 60675013).

References

Banfield, R.E., Hall, L.O., Bowyer, K.W., Kegelmeyer, W.P., 2007. A comparison of decision tree ensemble creation techniques. IEEE Trans. Pattern Anal. Mach. Intell. 29 (1), 173–180.

Blake, C.L., Merz, C.J., 1998. UCI repository of machine learning databases, URL available from: http://www.ics.uci.edu/~mlearn/MLRepository.html.

Breiman, L., 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140.

Breiman, L., 1998. Arcing classifiers. Ann. Statist. 26 (3), 801–849.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.

Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Chapman & Hall, New Work.

Chandra, A., Yao, X., 2006. Evolving hybrid ensembles of learning machines for better generalization. Neurocomputing 69 (7–9), 686–700.

Dietterich, T.G., 2000. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization. Mach. Learn. 40 (2), 139–157.

Drucker, H., Cortes, C., 1996. Boosting decision trees. Neural Inform. Proc. Sys. 8, 479–485.

Efron, B., Tibshirani, R.J., 1993. An Introduction to the Bootstrap. Chapman & Hall, New York.

Fleiss, J.L., Levin, B., Paik, M.C., 1981. Statistical Methods for Rates and Proportions. Wiley, New York.

Freund, Y., Schapire, R.E.. 1996. Experiments with a new boosting algorithm. In: Proceedings of the 13th International Conference on Machine Learning. Bari, Italy, 148–156.

Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. System Sci. 55 (1), 119–139.

Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Ann. Statist. 29 (5), 1189–1232.

Friedman, J.H., 2002. Stochastic gradient boosting. Comput. Statist. Data Anal. 38 (4), 367–378.

Friedman, J., Hastie, T., Tibshirani, R., 2000. Additive logistic regression: a statistical view of boosting. Ann. Statist. 28 (2), 337–407.

Hansen, L., Salamon, P., 1990. Neural networks ensembles. IEEE Trans. Pattern Anal. Mach. Intell. 12 (10), 993–1001.

Kearns, M.J., Vazirani, U.V., 1994. An Introduction to Computational Learning Theory. MIT Press, Cambridge.

Latinne, P., Debeir, O., Decaestecker, C., 2002. Combining different methods and number of weak decision trees. Pattern Anal. Appl. 5 (2), 201–209.

Maclin, R., Optiz, D., 1997. An empirical evaluation of bagging and boosting. In: Proceedings of the 14th National Conference on Artificial Intelligence. AAAI Press, RI, pp. 546–551.

Margineantu, D., Dietterich, T., 1997. Pruning adaptive boosting. In: Proceedings of the 14th International Conference on Machine Learning. Morgan, Kafmann, pp. 211–218.

Optiz, D., Maclin, R., 1999. Popular ensemble methods: an empirical study. J. Artif. Intell. Res. 11, 169–198.

Optiz, D.W., Shavlik, J.W., 1996. Generating accurate and diverse members of a neural-network ensemble. Neural Inform. Proc. System 8, 535–541.

Rätch, G., Onoda, T., Müller, K.R., 2001. Soft margins for Adaboost. Mach. Learn. 42 (3), 287–320.

Rodríguez, J.J., Kuncheva, L.I., Alonso, C.J., 2006. Rotation forest: a new classifier ensemble method. IEEE Trans. Pattern Anal. Mach. Intell. 28 (10), 1619–1630.

Schapire, R.E., 1990. The strength of weak learnability. Mach. Learn. 5 (2), 197–227.

Schapire, R.E., Singer, Y., 1999. Improved boosting algorithms using confidence-rated predictions. Mach. Learn. 37 (3), 297–336.

Tresp, V., 2001. Committee machines. In: Hu, Y., Hwang, H.-N. (Eds.), Handbook for Neural Network Signal Processing. CRC Press, Boca Raton, FL.

Wolpert, D.H., 1992. Stacked generalization. Neural Networks 5 (2), 241–259.