Pattern Recognition 45 (2012) 4451–4465

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A noise-detection based AdaBoost algorithm for mislabeled data

Jingjing Cao, Sam Kwong*, Ran Wang

Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong

ARTICLE INFO

Article history:
Received 14 September 2011
Received in revised form
16 March 2012
Accepted 5 May 2012
Available online 16 May 2012

Keywords:
Pattern recognition
Ensemble learning
AdaBoost
k-NN
EM

ABSTRACT

Noise sensitivity is known as a key related issue of AdaBoost algorithm. Previous works exhibit that AdaBoost is prone to be overfitting in dealing with the noisy data sets due to its consistent high weights assignment on hard-to-learn instances (mislabeled instances or outliers). In this paper, a new boosting approach, named noise-detection based AdaBoost (ND-AdaBoost), is exploited to combine classifiers by emphasizing on training misclassified noisy instances and correctly classified non-noisy instances. Specifically, the algorithm is designed by integrating a noise-detection based loss function into AdaBoost to adjust the weight distribution at each iteration. A *k*-nearest-neighbor (*k*-NN) and an expectation maximization (EM) based evaluation criteria are both constructed to detect noisy instances. Further, a regeneration condition is presented and analyzed to control the ensemble training error bound of the proposed algorithm which provides theoretical support. Finally, we conduct some experiments on selected binary UCI benchmark data sets and demonstrate that the proposed algorithm is more robust than standard and other types of AdaBoost for noisy data sets.

1. Introduction

AdaBoost [1–3] is one of the most popular techniques for generating ensembles due to its adaptability and simplicity. In the past few decades, AdaBoost has been successfully extended to many fields such as cost-sensitive classification [4,5], semi-supervised learning [6], tracking [7] and network intrusion detection [8]. The main idea of AdaBoost is to construct a succession of weak learners by using different training sets that are derived from resampling the original data. Through a weighted vote, these learners are combined to predict the class label of a new test instance. Normally, the performance of a weak learner is slightly better than random guessing [9]. The weak learner that used in the ensemble is named as base classifier or component classifier.

However, AdaBoost tends to be overfitting when the number of combined classifiers increases. Some researchers attributed this failure of AdaBoost to the high proportion of noisy instances [10,11]. In [10], Rätsch et al. defined three conditions to identify noisy data: (1) overlapping class probability distributions, (2) outliers and (3) mislabeled instances. It should be noted that our work only discusses noisy instances with mislabeled property. Mislabeled instances typically refer to those instances inconsistent with most of their surrounding neighbors' class labels. Dietterich [11] designed an experimental test to prove the poor generalization of AdaBoost with C4.5 by adding artificial noise. He explained that the mislabeled instances would possibly be assigned to higher weights, which gave rise to unsatisfactory performance of AdaBoost.

By analyzing the inner impelling force of AdaBoost, one may notice that essentially it aims to minimize an exponential loss function [12] sequentially. In detail, it puts emphasis on penalizing misclassified instances by giving incremental weights whereas assigning lessened weights to correctly classified instances for the next iteration. In this way, AdaBoost will only focus on punishing the misclassified instances whereas ignore their mislabeled property, which leads to the noise sensitivity of AdaBoost.

Therefore, in this paper, a noise-detection based AdaBoost algorithm (ND-AdaBoost), associated with the mislabeled properties of instances, is proposed to address the noise sensitivity and overfitting problem. The main contributions of this paper are as follows.

(1) Four types of instances with respect to noise and class label decisions, which are different from conventional concern on taxonomy of misclassified and correctly classified instances, are introduced. More specifically, they are correctly classified noisy instances, misclassified noisy instances, misclassified non-noisy instances and correctly classified non-noisy instances. This division is in line with the assumption that the probability of mislabeled instances being misclassified should be as high as possible, while the correctly labeled instances are expected to be classified correctly.

* Corresponding author. Tel.: +852 34427704; fax: +852 34420503.
E-mail address: cssamk@cityu.edu.hk (S. Kwong).

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.05.002

(2) A revised exponential loss function is proposed by considering these types of instances. At each iteration, a noise label determined by a noise detection function is assigned to each instance. With the new loss function, we aim to minimize the optimization objective by assigning less weights to misclassified noisy instances and correctly classified non-noisy instances. To identify noisy data, both EM and k-NN based functions are employed to test noise labeling effects under different detection methods.

(3) In order to guarantee the generalization ability of the proposed method, a new regeneration condition based on the analysis of empirical margin error bound of ND-AdaBoost is developed, so as to control the bound of the proposed algorithm within a reasonable range.

The performance of noise-detection based AdaBoost algorithm is examined through experiments on 13 binary data sets from UCI repository [13]. Experimental results show that the proposed algorithm outperforms other boosting methods under noisy environment.

2. Related work

For decades, researchers have made different modifications on AdaBoost technique to handle its noisy detrimental effect through two directions: (1) revising the optimization objective (loss function) and rebuilding the weight updating mechanism according to the corresponding loss function; (2) limiting the incremental weight update of the noisy instance or discarding them directly. Through these methods, the disturbance originated from mistrust instances could be minimized, and the noise tolerance of the ensemble model could be improved.

Regarding the first direction, many techniques have been employed. Depending on the statistical interpretation of AdaBoost, LogitBoost [14] utilized the additive logistic regression model function to replace the original loss function. However, it often suffers numerical problems caused by computing the regression variable. A generalized version of traditional AdaBoost is called Real AdaBoost [14,15]. It calculates the class probability to construct better real-valued output of weak learners. In essence, Real AdaBoost employs a log-odds ratio to replace the exponential loss function. With this modification, Real AdaBoost can converge quicker than AdaBoost but is also sensitive to outliers and mislabeled data [16]. Hastie and Tibshirani [14] presented an improved version of Real AdaBoost by utilizing adaptive Newton steps, which was similar to LogitBoost algorithm, to minimize the loss function. The empirical evidence implies that Gentle AdaBoost outperforms Real AdaBoost in terms of noisy data but has similar performance on regular data. Since weak learner may bias on training the data that have been correctly classified with high margin, Modest AdaBoost [17] was proposed to revise the loss function by focusing on decreasing this impact of base classifier [18]. This modification improves the generalization capability and relieves the overfitting problem of AdaBoost to some extent. MadaBoost was proposed by [19] with the aim to modify the reweighting scheme of AdaBoost. In the literature [19], Carlos et al. proved that one version of MadaBoost kept an adaptive boosting property. However, in the framework of MadaBoost, the corresponding advantages γ_i of the weak hypotheses ($\gamma_i \overset{\text{def}}{=} 1/2 - \epsilon_i$, ϵ_i denote the errors at iteration i) are monotonically decreasing and its boosting speed is slower than AdaBoost [16]. Rätsch et al. [10] claimed that AdaBoost intended to overfit on data set with high noise level. Inspired by the margin theory of SVM, a revised version of AdaBoost was introduced by embedding soft-margin into the algorithm AdaBoost_{Reg}. In this case, the noise effect could be mitigated by controlling the influence of an instance on ensemble classifiers. In summary, by utilizing various loss functions, these methods can perform well on noisy data.

However, Gao et al. [20] pointed out that the modified loss functions were fixed and were independent of the noisy or noisy-free property of input instances. Thus, they took into account the filter procedure to filter the noisy instances. Likewise, as for the second trend, Nikunj [21] proposed a revised Aveboost2 by averaging the current and the previous distributions to generate new base classifiers' distribution. Nicolas [22,23] employed the distribution generated by boosting to build a supervised projection of the original data to train the next classifier. However, Nikunj [21] and Nicolas [22,23] only emphasized on reducing the weights of the misclassified patterns while without considering the noise issue. Gao et al. [24] also proposed a weighted k-NN algorithm to identify and removed some suspect instances. But editing suspect instances will shrink the size of the training set. Additionally, it is probable that the mislabeled instances are maintained while the correctly labeled instances are removed.

Based on the above-mentioned discussions, we propose a noise-detection based method by modifying the loss function. It integrates the advantages of these two research directions while restricts the weight of instance simultaneously.

Furthermore, when dealing with noisy data, another major problem is how to detect noisy instances. It is found that most of the previous works are related to data pruning, but we are in favor of utilizing instance weighting approach instead of editing. For example, Rätsch et al. [10] described that mistrust instances were those which are highly influential to the decision. They used the average weight of an instance based upon the assumption that a difficultly classified instance probably has high average weight. Rebbapragada et al. [25] proposed a method named pairwise expectation maximization (PWEM) to produce instance weight. They conducted some experiments to show that instance weighting performed better than instance editing.

Different from selecting subset of instances to generate new weight distribution, another groups of Boosting algorithms emphasize on choosing subset of features to construct ensemble of classifiers. Random subspace method (RSM) [26] is a standard method to randomly choose a subspace from the original feature space, which could be composed from any base classifier. In [27], Satoshi et al. empirically showed that combining RSM and AdaBoost algorithm had less generalization error than using RSM and AdaBoost, respectively. Additionally, Nicolas et al. [28] improved the performance of the RSM based AdaBoost by concerning the discriminate information among subspaces. Recently, Nanni et al. [29] designed a Reduced Reward-punishment editing strategy to construct different subspaces used in feature transform based ensemble approaches. In our experimental studies, we also compare our approach with the RSM+AdaBoost method since it is more robust than AdaBoost in the presence of noise [28].

The rest of this paper is arranged as follows: the related knowledge about AdaBoost is introduced in Section 3; the motivation and the procedure of the proposed algorithm are provided in Section 4 which are associated with some related analysis; the experimental comparisons on 13 real world binary data sets are presented and analyzed in Section 5; discussions and conclusions of the paper are finally given in Section 6.

3. Framework of AdaBoost algorithm

Since our work is an extension of AdaBoost approach, preliminary knowledge of AdaBoost is firstly introduced in this section. Suppose we have a two class supervised learning task. Let the n-th training instance denoted as $z_n = (x_n, y_n), n = 1, \ldots, N$,

$x_n \in X$ and y_n is class label of x_n. Let H be a set of component classifiers: $H = \{h_t(x_n) : X \to \{-1,+1\}, t = 1,\dots,T\}$. The pseudo-code of basic AdaBoost algorithm for binary classification is presented in Algorithm 1.

Algorithm 1. AdaBoost.

Input: A set of labeled training instances:
$S = \{(x_n, y_n),\ n = 1,2,\dots,N\}$, where $y_n \in \{-1,1\}$, maximum number of base classifiers T.
Initialization: $w_n^{(1)} = 1/N,\ n = 1,2,\dots,N;\ t = 1$.
while $t \le T$ **do**
  Resampling a training subset $TR_t = \{(x_n, y_n),\ n = 1,2,\dots,N\}$ with replacement;
  Train the TR_t using a base learning algorithm to obtain weak learner h_t;
  Calculate the weight error of training instances $\epsilon_t = \sum_{n=1}^{N} w_n^{(t)} I(h_t(x_n) \ne y_n)$;
  if $\epsilon_t > 0.5$ *or* $\epsilon_t = 0$ **then**
    Generate uniformly distributed weights $w_n^{(t)} = 1/N$, $n = 1,2,\dots,N$, continue;
  Calculate $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$;
  Update the weights of training instances with $w_n^{(t+1)} = w_n^{(t)} \exp(-\alpha_t h_t(x_n) y_n)$;
  Normalization : $w_n^{(t+1)} = w_n^{(t+1)} / \sum_{n=1}^{N} w_n^{(t+1)}$;
  $t = t+1$;
Final classifier h_f is obtained by weighted majority voting for a testing instance x_n^* : $h_f(x_n^*) = sign(\sum_{t=1}^{T} \alpha_t h_t(x_n^*))$.

3.1. Loss function

The goal of AdaBoost algorithm is to construct an optimal classifier by minimizing an exponential loss function [30]. The loss function, which is a key concept in learning theory, is also called cost function, or error function in some cases. Considering the loss function of AdaBoost, we have the following definition.

Definition 1. Exponential loss function that applied in AdaBoost is defined as follows:

$$L = \sum_{n=1}^{N} \exp(-y_n F_T(x_n)) = \sum_{n=1}^{N} \exp[-y_n(\alpha_T h_T(x_n) + F_{T-1}(x_n))], \quad (1)$$

where α_t is the weight of each base classifier h_t, and $F_{T-1}(x_n)$ is the sum of the learners of instance x_n from the previous iterations:

$$F_{T-1}(x_n) = \sum_{t=1}^{T-1} \alpha_t h_t(x_n).$$

Then, the weight $w_n^{(T)}$ is introduced as a constant for the T-th iteration since it is not related to either α_T or $h_T(x_n)$. So (1) could be changed as

$$L = \sum_{n=1}^{N} w_n^{(T)} \exp[-y_n \alpha_T h_T(x_n)], \quad (2)$$

where $w_n^{(T)} = \exp(-y_n F_{T-1}(x_n))$.

3.2. Margin and ensemble training error bound

Margin theory explains the inner motivation which makes AdaBoost successful, where AdaBoost enlarges the margin to obtain a good generalization capability [10,31].

Definition 2. (Rätsch et al. [10]). Margin of instance $z_n = (x_n, y_n)$ is defined as follows:

$$\rho(z_n, T) = y_n f_T(x_n) = y_n \frac{\sum_{t=1}^{T} \alpha_t h_t(x_n)}{\sum_{t=1}^{T} \alpha_t}, \quad (3)$$

where $f_T(x_n) = F_T(x_n) / \sum_{t=1}^{T} \alpha_t$.

It can be seen that $\rho > 0$ represents that the instance has been correctly classified; otherwise, the instance has been misclassified.

In this paper, the expectation of margin error is considered as follows:

$$\varepsilon = \widehat{E}\{I(y f_T(x) \le \theta)\} = \frac{1}{N} \sum_{n=1}^{N} I[y_n f_T(x_n) \le \theta]. \quad (4)$$

Theorem 1 (*Schapire et al. [31]*). *Let ε be the expectation of margin error of the AdaBoost algorithm, let $\epsilon_t (t = 1,2,\dots,T)$ be the weighted training errors of the base classifiers $\epsilon_t = \sum_{n=1}^{N} w_n^{(t)} I[h_t(x_n) \ne y_n]$. Then for any θ, we have that*

$$\varepsilon \le \prod_{t=1}^{T} \sqrt{4\epsilon_t^{1-\theta}(1-\epsilon_t)^{1+\theta}}. \quad (5)$$

For $\theta = 0$, ε is equal to the proportion of misclassified instances, i.e., the ensemble training error $\epsilon = \sum_{n=1}^{N} I[y_n f_T(x_n) \le 0] / N \le \prod_{t=1}^{T} \sqrt{4\epsilon_t(1-\epsilon_t)}$. This bound indicates that if we can continually find weak learners with errors less than a half, then the ensemble training error of the final classifier will decrease exponentially.

3.3. Regeneration condition

It is worth noting that to obey the optimization rule, the weight of each classifier should satisfy the condition that $\alpha_t \ge 0$. Thus, the training errors $\epsilon_t (t = 1,2,\dots,T)$ of all the component classifiers are limited to the range $(0, 1/2]$. Once a condition dissatisfied classifier is generated, two alternative methods could be employed: one is by stopping the algorithm and the other is by regenerating a uniform weight for each instance. In this paper, we use the latter mechanism and add a new condition to control the generalization bound of the proposed approach.

4. Noise-detection based AdaBoost

Before introducing our proposed algorithm, two important assumptions that related to the noise issue are presented as follows:

- **Assumption 1**: An optimal classifier is supposed to be able to correctly classify the clean instances, i.e., non-noisy instances.
- **Assumption 2**: An optimal classifier is supposed to be able to misclassify the mislabeled instances, i.e., noisy instances.

4.1. New instances type categorization

In boosting, only misclassified instances are assigned higher weights for the next generation. Such scheme ignores the noisy impact caused by the instances. If a misclassified instance is noisy, its weight enhancement will force the classifier to emphasize on classifying mislabeled instance correctly. In this case, the classifier tends to be overfitting on training data. On the contrary, according to Assumption 2, a noisy instance is supposed to be misclassified, while in real application, it may be correctly classified by a non-perfect classifier. Then, its weight reduction will further decrease the chance of it being misclassified. Thus, in

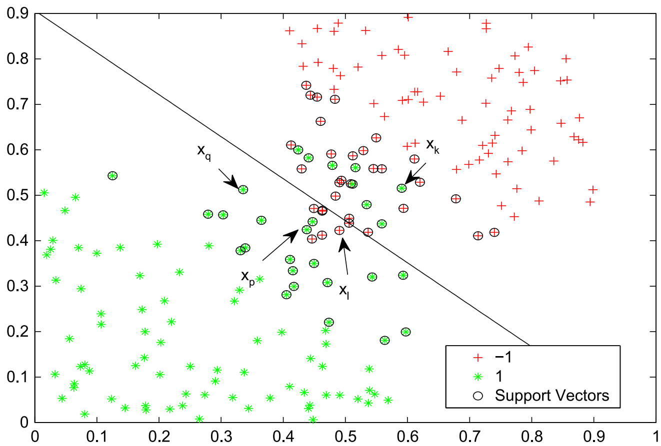

Fig. 1. Four types of instances with respect to noise factor and classified label.

this paper, we aim to propose a loss function which can cover the above-discussed issues. Four types of instances are introduced below, which will be the basis for discussing the proposed work:

(1) Instances that are correctly classified and regarded as noisy data, denoted as $x_p \in X_p, 1 \le p \le N_p$;

(2) Instances that are correctly classified and regarded as non-noisy data, denoted as $x_q \in X_q, 1 \le q \le N_q$;

(3) Instances that are misclassified and regarded as noisy data, denoted as $x_k \in X_k, 1 \le k \le N_k$;

(4) Instances that are misclassified and regarded as non-noisy data, denoted as $x_l \in X_l, 1 \le l \le N_l$;

where X_p, X_q, X_k, X_l represent the sets according to the types of instances. Additionally, N_p, N_q, N_k, N_l denote the total element number of the sets X_p, X_q, X_k, X_l.

Fig. 1 shows an example of the four types of instances. The binary data are classified by linear support vector machine (SVM) where point "○" denotes the support vector. Points "+" and "*" represent the training data with class label -1 and $+1$, respectively. These instances are defined to later analyze the effectiveness of the proposed noise-detection based loss function. The noisy label for each instance may change at different iteration due to the various surrounding environments that are generated by the resampling scheme. In Section 4.3, two noise-detection functions will be employed to detect whether the instances are noisy data.

4.2. Noise-detection based loss function and margin

Here, we will explain our algorithm based on the previous definitions of loss function and the margin of instance. After that, an analysis of the proposed definitions will be provided.

Definition 3. Noise-detection based margin of instance $z_n = (x_n, y_n)$:

$$\hat{\rho}(z_n, T) = \frac{y_n \sum_{t=1}^{T} \alpha_t h_t(x_n)\phi_t(x_n)}{\sum_{t=1}^{T} \alpha_t}, \tag{6}$$

where $\phi_t(x_n) = \mathrm{sgn}(\overline{\mu}_t - \mu_t(x_n))$ is a noise-detection function that determines whether instance x_n is noisy. $\phi_t(x_n) \in \{-1, 1\}$ where $\phi_t(x_n) = -1$ denotes that data x_n is regarded as a noisy instance at the t-th iteration.

With function $\phi_t(x_n)$, $\mu_t(x_n)$ is the probability of the event that instance x_n is noisy. In other words, it represents the confidence of each instance being a noisy instance. $\overline{\mu}_t = \sum_{n=1}^{N} \mu_t(x_n)/N$ denotes the mean value among all the training instances at the t-th iteration. Thus the new loss function is defined as follows:

Definition 4. Noise-detection based loss function

$$\hat{L} = \sum_{n=1}^{N} \hat{w}_n^{(T)} \exp[-y_n \alpha_T h_T(x_n)\phi_T(x_n)], \tag{7}$$

where $\hat{w}_n^{(T)} = \exp(-y_n \sum_{t=1}^{T-1} \alpha_t h_t(x_n)\phi_t(x_n))$.

With Definition 4, we divide the formula into two items depending on the hypothesis classification labels in current iteration t:

$$\hat{L} = \sum_{y_n h_t \phi_t = -1} \hat{w}_n^{(t)} \exp(\alpha_t) + \sum_{y_n h_t \phi_t = +1} \hat{w}_n^{(t)} \exp(-\alpha_t). \tag{8}$$

In this formula, h_t and ϕ_t are short for $h_t(x_n)$ and $\phi_t(x_n)$, respectively. Then, for a fixed value of α_t, the cost will be optimized with respect to $h_t(x_n)$ and $\phi_t(x_n)$. Let $\delta_t = \sum_{n=1}^{N} \hat{w}_n^{(t)} I(y_n \neq (h_t(x_n)\phi_t(x_n))) = \sum_{y_n h_t \phi_t = -1} \hat{w}_n^{(t)}$ and $1-\delta_t = \sum_{n=1}^{N} \hat{w}_n^{(t)} I(y_n = (h_t(x_n)\phi_t(x_n))) = \sum_{y_n h_t \phi_t = +1} \hat{w}_n^{(t)}$, according to [32], by using the binary nature of the component classifier h_t, the optimized problem (8) is changed to an equivalent objective function as follows:

$$\tilde{L} = \delta_t e^{\alpha_t} + (1-\delta_t) e^{-\alpha_t}. \tag{9}$$

Taking the derivative with respect to α_t and equating to zero, then we get

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\delta_t}{\delta_t}\right). \tag{10}$$

The weights for the next step are computed by the following equation:

$$\hat{w}_n^{(t+1)} = \hat{w}_n^{(t)} \exp(-\hat{\rho}(z_n, t)). \tag{11}$$

In the following, we analyze the interpretation of the new loss function when dealing with misclassified and correctly classified instances:

(a) Instance x_n is misclassified. Then we have $y_n h_t(x_n) = -1$ and Eq. (8) is reduced as follows:

$$\hat{L} = \sum_{\phi_t = +1} \hat{w}_n^{(t)} \exp(\alpha_t) + \sum_{\phi_t = -1} \hat{w}_n^{(t)} \exp(-\alpha_t). \tag{12}$$

Thus, we have $x_n \in X_k \cup X_l$, and suppose that $x_k \in X_k$ is noisy while $x_l \in X_l$ is non-noisy, thus we assume that $\mu(x_l) < \overline{\mu}_t < \mu(x_k)$, which leads to $-1 = \phi_t(x_k) < 0 < \phi_t(x_l) = 1$.

It implies that for the misclassified data, we would rather make noisy data be misclassified than non-noisy data. The noisier the incorrectly classified data is, the less value the loss function has.

(b) Instance x_n is correctly classified. Then we have $y_n h_t(x_n) = +1$, and Eq. (8) is transformed as follows:

$$\hat{L} = \sum_{\phi_t = -1} \hat{w}_n^{(t)} \exp(\alpha_t) + \sum_{\phi_t = +1} \hat{w}_n^{(t)} \exp(-\alpha_t). \tag{13}$$

Suppose that $x_p \in X_p$ is noisy while $x_q \in X_q$ is not noisy, thus we assume that $\mu(x_q) < \overline{\mu}_t < \mu(x_p)$, which leads to $-1 = \phi_t(x_p) < 0 < \phi_t(x_q) = 1$. From Eq. (13), it can be concluded that for the correctly classified data, non-noisy data tend to minimize the loss function, i.e., the number of non-noisy data is preferred to be correctly classified as many as possible.

4.3. Noise-detection criterion construction

The noise detection criterion is used to detect the noisy label of each training instance. Designing an appropriate noise detection function $\phi_t(\cdot)$ is crucial in our proposed algorithm. Since we use formula $\phi_t(x_n) = \mathrm{sgn}(\overline{\mu}_t - \mu_t(x_n))$ mentioned in Definition 3 to determine the noisy data unitedly, the key issue will become how to select proper confidence probability criterion $\mu_t(\cdot)$ instead. Obviously, there are various approaches to evaluate the weight of noise instance depending on the available information. In this work, the k-nearest neighbor (k-NN) and the expectation maximization (EM) algorithms are both employed so as to assess the prediction performance of the proposed algorithm based on different noise-detection criteria.

The k-NN algorithm [33] is well known as a non-parametric classification method. It is simple to implement and easy to interpret. The k-NN based criterion is proposed as follows:

$$\mu_t^{knn}(z_i) = \sum_{z_j \in K_i^t} I(h_t(x_j) \neq y_j)/k, \tag{14}$$

where K_i^t represents a set including the k nearest neighbors $z_j = (x_j, y_j)$ of z_i at the t-th iteration, $h_t(x_j)$ is the class label predicted by base classifier h_t.

As for EM based criteria, we directly use the pair-wise expectation maximization (PWEM) method [25] to calculate the probability $P(c|z_i)$ that instance z_i belonging to class c. Then we propose the definition of confidence probability criterion as follows:

$$\mu_t^{EM}(z_i) = P(c^*|z_i) - P(c^{**}|z_i), \tag{15}$$

where $c^* = \arg\max\{P(c|z_i)|c = 1,\ldots,C\}$, and $c^{**} = \arg\max\{P(c|z_i)|c = 1,\ldots,C; c \neq c^*\}$. c^{**} is the class with the second largest weight of instance z_i.

4.4. Regeneration condition

Normally, the regeneration condition of AdaBoost is $0 < \epsilon_t \leq 0.5$, $t = 1,\ldots,T$. In this section, we present that how a new regeneration condition could be added to ensure good generalization performance of our proposed algorithm.

Theorem 2. *Let ε' be the expectation of margin error of the proposed algorithm; let $\delta_t (t = 1,2,\ldots,T)$ be the weighted training error with noisy-detection function of the ensemble classifiers $\delta_t = \sum_{n=1}^N w_n^{(t)} I(y_n \neq (h_t(x_n)\phi_t(x_n)))$, then we have*

$$\varepsilon' \leq \prod_{t=1}^T \sqrt{4\delta_t^{1-\theta}(1-\delta_t)^{1+\theta}}. \tag{16}$$

Proof. We first define $M_t = \sum_{n=1}^N w_n^{(t)} e^{-y_n h_t(x_n)\phi_t(x_n)\alpha_t}$. Due to the binary nature of $y_n, h_t(x_n), \phi_t(x_n) \in \{-1,+1\}$, it is easy to deduce that the following two equations are equivalent to each other:

$$y_n h_t(x_n)\phi_t(x_n) = +1(-1) \Leftrightarrow I(y_n = (h_t(x_n)\phi_t(x_n))) = 1(0)$$

and

$$y_n h_t(x_n)\phi_t(x_n) = -1(+1) \Leftrightarrow I(y_n \neq (h_t(x_n)\phi_t(x_n))) = 1(0).$$

Further, since α_t relies on neither $w_n^{(t)}$ nor $h_t(x_n)\phi_t(x_n)$, it can be considered as a constant only associated with iteration t. Due to the definition of $w_n^{(t)}$, we have $\sum_{n=1}^N w_n^{(t)} = 1$, then $1-\delta_t = \sum_{n=1}^N w_n^{(t)} I(y_n = (h_t(x_n)\phi_t(x_n)))$. Thus, M_t could be divide into the following two parts:

$$\begin{aligned} M_t &= \sum_{n=1}^N w_n^{(t)} e^{-\alpha_t} I(y_n = (h_t(x_n)\phi_t(x_n))) + \sum_{n=1}^N w_n^{(t)} e^{\alpha_t} I(y_n \neq (h_t(x_n)\phi_t(x_n))) \\ &= (1-\delta_t)e^{-\alpha_t} + \delta_t e^{\alpha_t}. \end{aligned} \tag{17}$$

Then, due to Eq. (10), α_t is selected as $\alpha_t = \frac{1}{2}\ln(1-\delta_t)/\delta_t$, thus,

$$M_t = 2\sqrt{\delta_t(1-\delta_t)}. \tag{18}$$

Recall the definitions of $f_T(x_n)$ and ε in Eqs. (3) and (4), the expectation of margin error of our algorithm is defined as follows:

$$\varepsilon' = \widehat{E}\{I(yf'_T(x) \leq \theta)\} = \frac{1}{N}\sum_{n=1}^N (y_n f'_T(x_n) \leq \theta), \tag{19}$$

where

$$f'_T(x_n) = \frac{\sum_{t=1}^T \alpha_t h_t(x_n)\phi_t(x)}{\sum_{t=1}^T \alpha_t}.$$

Thus, we have

$$\begin{aligned} yf'_T(x) \leq \theta &\Rightarrow y\sum_{t=1}^T \alpha_t h_t(x)\phi_t(x) \leq \theta \sum_{t=1}^T \alpha_t \\ &\Rightarrow \exp\left(-y\sum_{t=1}^T \alpha_t h_t(x)\phi_t(x) + \theta\sum_{t=1}^T \alpha_t\right) \geq 1 \\ &\Rightarrow \exp\left(-y\sum_{t=1}^T \alpha_t h_t(x)\phi_t(x) + \theta\sum_{t=1}^T \alpha_t\right) \geq I(yf'_T(x) \leq \theta). \end{aligned} \tag{20}$$

As $I(yf'_T(x) \leq \theta) \geq 0$ and θ is independent to other factors, we can get the following induction according to inequality (20):

$$\begin{aligned} \varepsilon' &= \widehat{E}\{I(yf'_T(x) \leq \theta)\} \leq \widehat{E}\{e^{-y\sum_{t=1}^T \alpha_t h_t(x)\phi_t(x) + \theta\sum_{t=1}^T \alpha_t}\} \\ &= \frac{1}{N} e^{\theta\sum_{t=1}^T \alpha_t} \sum_{n=1}^N e^{-y_n \sum_{t=1}^T \alpha_t h_t(x_n)\phi_t(x_n)} \\ &= e^{\theta\sum_{t=1}^T \alpha_t}\left(\prod_{t=1}^T M_t\right)\left(\sum_{n=1}^N w_n^{T+1}\right) \\ &= e^{\theta\sum_{t=1}^T \alpha_t}\left(\prod\nolimits_{t=1}^T M_t\right), = \prod_{t=1}^T \sqrt{4\delta_t^{1-\theta}(1-\delta_t)^{1+\theta}}, \end{aligned}$$

where

$$\begin{aligned} w_n^{(T+1)} &= \frac{w_n^{(T)}\exp(-y_n\alpha_T h_T(x_n)\phi_T(x_n))}{M_T} \\ &= \frac{\exp(-\sum_{t=1}^T \alpha_t h_t(x_n) y_n \phi_t(x_n))}{N\prod_{t=1}^T M_t}. \qquad \square \end{aligned} \tag{21}$$

Theorem 2 is proved based on Theorem 1 [31] and shows a similar result as Theorem 1. Then we also have the ensemble training error $\epsilon' = \sum_{n=1}^N I[y_n f_T(x_n) \leq 0]/N \leq \prod_{t=1}^T \sqrt{4\delta_t(1-\delta_t)}$. Next, we analyze the relationship between ϵ_t and δ_t for $t = 1,2,\ldots,T$ when $\theta = 0$.

$$\begin{aligned} \delta_t &= \sum_{n=1}^N w_n^{(t)} I[y_n \neq (h_t(x_n)\phi_t(x_n))] \\ &= \sum_{n \in C_t} w_n^{(t)} I(y_n \neq h_t(x_n)) + \sum_{n \in N_t} w_n^{(t)} I(y_n = h_t(x_n)) \\ &= \epsilon_t + \sum_{n \in N_t} w_n^{(t)}[I(y_n = h_t(x_n)) - I(y_n \neq h_t(x_n))], \end{aligned} \tag{22}$$

where $C_t = \{n|\phi_t(x_n) = +1\}$ and $N_t = \{n|\phi_t(x_n) = -1\}$. Let $Bound(t) = \sum_{n \in N_t} w_n^{(t)} I(y_n = h_t(x_n)) - \sum_{n \in N_t} w_n^{(t)} I(y_n \neq h_t(x_n))$ for each iteration, if condition

$$Bound(t) < 0 \tag{23}$$

is satisfied, then $\delta_t < \epsilon_t$. According to inequalities (5) and (16), one could guarantee that the ensemble training error bound of the proposed algorithm is smaller than AdaBoost, i.e., $\epsilon' < \epsilon$.

Suppose that we add this regeneration condition to the algorithm, it would not be difficult to find out that Bagging-like algorithm could be considered as a special case of this algorithm. In other words, once inequality (23) is consistently dissatisfied at each iteration, the weight distribution will keep being uniform and AdaBoost algorithm becomes Bagging-like algorithm. However, it will loosen the adaptive ability of AdaBoost, and limit the diversity of weight distribution as well. To avoid the limitation of this condition, we replace formula (23) with the following condition:

$$Bound(t) < \frac{1}{t-1}\sum_{m=1}^{t-1}\beta(m)Bound(m), \tag{24}$$

where $\beta(m)=\alpha_m/\sum_{l=1}^{t-1}\alpha_l$ is regarded as the weight of $Bound(m)$.

This equation denotes that the algorithm will regenerate uniform weight once the $Bound(t)$ is smaller than the average value of previous weighted $Bound(m)$,$m=1,2,\ldots,t-1$. With this new modification, it controls the bound of the new boosting algorithm in an increasingly stable range as the number of training iteration grows.

Thus, the pseudo-code of ND-AdaBoost algorithm with two-class data set is represented in Algorithm 2.

Algorithm 2. Noise-detection based AdaBoost.

Input: A set of labeled training instances:
$S=\{(x_n,y_n),\ n=1,2,\ldots,N\}$, where $y_n\in\{-1,1\}$, maximum number of base classifiers: T.
Initialize $\widehat{w}_n^{(1)}=1/N, n=1,2,\ldots,N;\ t=1$.
while $t\le T$ **do**
  Resampling a training subset $TR_t=\{(x_n,y_n),\ n=1,2,\ldots,N'\}$ with replacement;
  Train the TR_t with base learning algorithm to obtain weak classifier h_t;
  Calculate the weight error of training instances $\delta_t=\sum_{y_nh_t(x_n)\phi_t(x_n)=-1}\widehat{w}_n^{(t)}$;
  Calculate the weight of $Bound(m)$:
  $\beta(m)=\alpha_m/\sum_{l=1}^{t-1}\alpha_l, m=1,\ldots,t-1$;
  If $\delta_t>0.5$ *or* $\delta_t=0$ *or* $Bound(t)<\frac{1}{t-1}\sum_{m=1}^{t-1}\beta(m)Bound(m)$ **then**
    Generate uniformly distributed weights $\widehat{w}_n^{(t)}=1/N$, $n=1,2,\ldots,N$, continue;
  Calculate $\alpha_t=\frac{1}{2}\ln(\frac{1-\delta_t}{\delta_t})$;
  Update the weights of training instances with
  $\widehat{w}_n^{(t+1)}=\widehat{w}_n^{(t)}\exp[-y_n\alpha_th_t(x_n)\phi_t(x_n))]$;
  where $\phi_t(x_n)\in\{+1,-1\}$, if -1,then x_n is noisy, otherwise x_n is non $-$ noisy.
  Normalization : $\widehat{w}_n^{(t+1)}=\widehat{w}_n^{(t+1)}/\sum_{n=1}^{N}\widehat{w}_n^{(t+1)}$;
  $t=t+1$;
Final classifier h_f obtained by weighted majority voting for a testing instance x_n^*, $h_f(x_n^*)=\mathrm{sign}(\sum_{t=1}^{T}\alpha_th_t(x_n^*))$

4.5. Update weight comparison between AdaBoost and ND-AdaBoost

Let $w_n^{(t+1)}$ and $\widehat{w}_n^{(t+1)}$ be the updated weights assigned on instance x_n after iteration t for AdaBoost and the proposed ND-AdaBoost. Then, the two weights are reshaped in the following equations:

$$w_n^{(t+1)}=\frac{w_n^{(t)}\exp[-\alpha_ty_nh_t(x_n)]}{Z_t}=\frac{w_n^{(t)}\Delta w_n^{(t)}}{Z_t},$$

$$\widehat{w}_n^{(t+1)}=\frac{\widehat{w}_n^{(t)}\exp[-\alpha_ty_nh_t(x_n)\phi_t(x_n)]}{\widehat{Z}_t}=\frac{\widehat{w}_n^{(t)}\Delta\widehat{w}_n^{(t)}}{\widehat{Z}_t},$$

where Z_t and $\widehat{Z}_t$ are the normalization factors for AdaBoost and ND-AdaBoost, respectively.

Suppose that four types of instances x_k,x_l,x_p,x_q are selected, which satisfied that $x_k\in X_k$, $x_l,\in X_l$, $x_p\in X_p$, $x_q\in X_q$,

*Case*1: (a) x_l is non-noisy and misclassified, i.e., $y_lh_t(x_l)=-1$ and $\mu_t(x_l)\to 0$, thus, we have $\phi_t(x_l)=+1$; (b) x_k is noisy and misclassified, i.e., $y_kh_t(x_k)=-1$ and $\mu_t(x_k)\to 1$, thus, we have $\phi_t(x_k)=-1$.

Then, the following order relationship is obtained:

$$-\alpha_t=\Delta\widehat{w}_k^{(t)}\le\Delta\widehat{w}_l^{(t)}=\Delta w_k^{(t)}=\Delta w_l^{(t)}=\alpha_t. \tag{25}$$

*Case*2: (a) x_q is non-noisy and correctly classified, i.e., $y_qh_t(x_q)=+1$ and $\mu_t(x_q)\to 0$, thus, we have $\phi_t(x_q)=+1$; (b) x_p is noisy and correctly classified, i.e., $y_ph_t(x_p)=+1$ and $\mu_t(x_p)\to 1$, thus, we have $\phi_t(x_p)=-1$.

Thus, we have the following order relationship:

$$-\alpha_t=\Delta w_p^{(t)}=\Delta w_q^{(t)}=\Delta\widehat{w}_q^{(t)}\le\Delta\widehat{w}_p^{(t)}=\alpha_t. \tag{26}$$

From Eqs. (25) and (26), it can be seen that for misclassified instances, AdaBoost assigns the same weights regardless of noise information. Instead, the proposed algorithm assigns larger weights on non-noisy instances than noisy instances. This difference decreases the probability that the next base classifier would consistently emphasize the training of these misclassified noisy instances. Contrarily, for correctly classified instances, non-noisy instances obtains less concern. This is because that non-noisy instances often show little effect on constructing the classifier. Therefore, the proposed algorithm focuses on training misclassified non-noisy instances and correctly classified noisy instances instead of misclassified instances.

5. Experiments

The motivation of ND-AdaBoost is to improve noise tolerance of conventional AdaBoost. Therefore, our experiments are primarily focusing on comparing classification accuracy/error at different noise levels. Two groups of experiments are examined to show the effectiveness of ND-AdaBoost algorithm in different aspects. The first group of experiments (Section 5.1) demonstrates the superiority of the noise-detection based method to benchmark algorithms in the GML AdaBoost Matlab Toolbox [34] with low noise level or without noise factor. Then Section 5.2 presents the performance of our proposed algorithm based on three different weak learners: k-nearest neighbor (k-NN), support vector machine (SVM) with Gaussian kernel and extreme learning machine (ELM) [35]. The experimental results clearly validate the advantages of the proposed algorithm. For all experiments, the test method of Dietterich [11] is used by adding a proportion r of the instances, and switching their class labels to the opposite class randomly. Then, we choose noise rates at 0%, 5% and 10% for the evaluations in the first part and 10%, 20% and 30% as the high noise rates in the latter experimental group. Thirteen binary data sets from the UCI Repository of Machine Learning [13] are employed and their information are listed in Table 1 and # Instance/Feature represent the number of instances/features for each data set. The data sets cover a wide range of application with small sample size (80) and large sample size (4601) cases. The parameter for SVM with Gaussian kernel is set as $\gamma=1$ in libsvm [36] environment.

For the k-NN method, which is a noise-detection criterion and also a weak learner in this paper, the choice of parameter k affects its performance significantly. When k is too small, the

Table 1
Dataset information.

Data	# Instance	# Feature
Australian	690	14
Bupa	345	6
Heart	270	13
Ionosphere	351	33
Sonar	208	60
Tic-Toc	958	9
Wpbc	198	33
Breast	699	9
German	1000	24
Hepatitis	80	19
Pima	768	8
Spam	4601	58
Wdbc	569	30

surrounding noisy information of each instance is not great enough to clearly distinguish it. However, when k is too large, the local learning capability of k-NN would drop. In this work, as for noise-detection function, k is set as 5 and we select 3-NN approach as one of the base classifiers.

5.1. Performance comparison of ND-AdaBoost with benchmark algorithms

Firstly, the GML AdaBoost Matlab Toolbox [34] is employed by using classification and regression tree (CART) [37] as weak learner. It aims at comparing the performance of ND-AdaBoost with benchmark algorithms: Real AdaBoost, Gentle AdaBoost and Modest AdaBoost. The experiments are conducted with the set up of examples 1 and 22 in the toolbox: (1) half of the data goes to

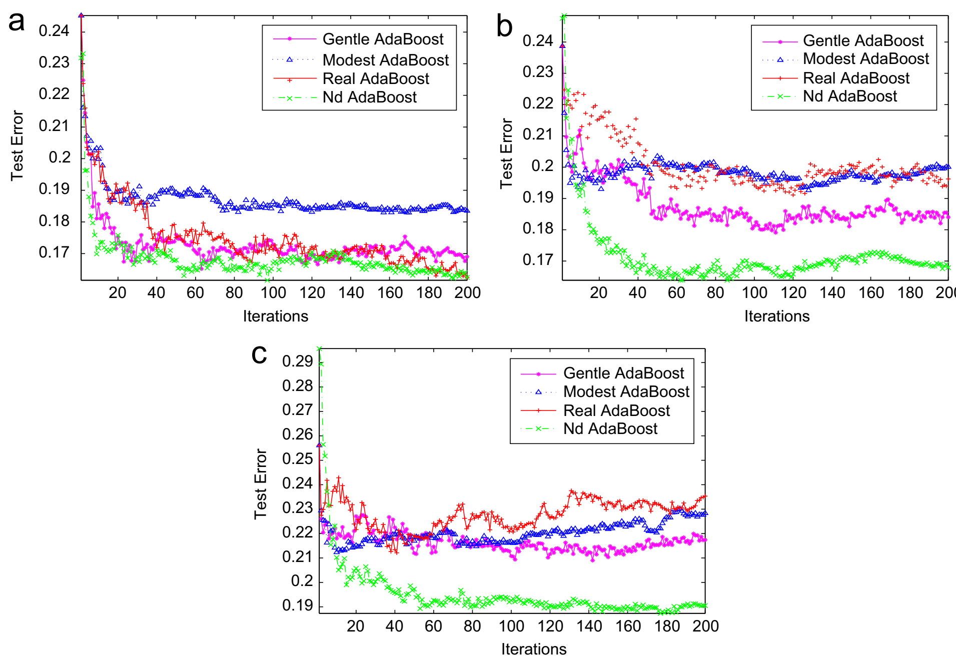

Fig. 2. Average testing error over all 12 data sets at low noise level. (a) 0% noise. (b) 5% noise. (c) 10% noise.

Table 2
Summary of average classification errors on test sets of UCI datasets.

Noise	0%				5%				10%			
Dataset	G	M	R	ND	G	M	R	ND	G	M	R	ND
Aust	.1942	**.1594**	.1884	.1797	.2232	.1623	.2116	**.1565**	.1855	.1884	.2493	**.1710**
Breast	.0401	**.0372**	.0401	**.0372**	.0659	.0458	.0774	**.0401**	.0974	.0487	.1318	**.0430**
Bupa	.2849	**.2500**	.3372	.2791	.3140	**.2907**	.3605	.2965	.3547	.4128	.3837	**.3488**
German	.3260	.2860	.3160	**.2420**	.3060	.2800	.3100	**.2600**	.3320	.2880	.3480	**.2400**
Heart	.2519	.2074	.2519	**.1556**	.2667	**.2000**	.2519	**.2000**	.2963	.2519	.3481	**.2370**
Spam	**.0530**	.0622	.0578	.0700	.0713	.0752	.0722	**.0700**	.1070	**.0709**	.1013	.0913
Iono	.0629	.0800	.0800	**.0571**	.1143	.1029	.1200	**.0800**	.1486	.1257	.1657	**.0914**
Pima	.3229	.2865	**.2760**	.2813	.3021	.2891	.3125	**.2786**	.3438	**.2917**	.3411	.2943
Sonar	.1635	.1635	**.1346**	.2308	**.1538**	.1731	.2404	.2115	**.2212**	.2404	.2692	.2981
Tic	.0167	.3111	**.0000**	.0877	.0752	.3549	**.0731**	.0939	.1378	.3528	.1315	**.0919**
Wdbc	.0493	.0458	**.0352**	.0599	.0634	.0634	**.0528**	.0669	.1021	.0951	**.0915**	.1021
Wpbc	**.2626**	.3131	.2323	.2727	**.2525**	.3636	.2727	.2727	.2828	.3737	**.2626**	.2929

"G" stands for Gentle AdaBoost, "M" for Modest AdaBoost, "R" for Real AdaBoost and "ND" for the proposed ND-AdaBoost; ".xxxx" is short for "0.xxxx". The bold data represent the lowest average testing error among all the classifiers at the corresponding noise level.

training set while the other to testing set. (2) The maximum number of iterations is set as $T=200$. For the Ionosphere dataset, we delete the first attribute since this attribute only has one value. However, it should be noted that this deletion may lead to different testing result among three standard AdaBoost algorithms. Furthermore, except Hepatitis dataset, the other 12 UCI datasets are utilized since Hepatitis dataset has too few instances which makes the given toolbox stops running automatically. As noise-detection function for ND-AdaBoost ensembles, we use k-NN based criterion that introduced in Eq. (14).

Fig. 2 presents the comparisons of the three standard methods and ND-AdaBoost in terms of the average testing error at noise levels 0%, 5% and 10%. The figure shows that AdaBoost in prone to overfit in the presence of noise. Though the performance of AdaBoost is similar to ND-AdaBoost and Gentle AdaBoost without noise, its testing error increases with the level of noise rising. On

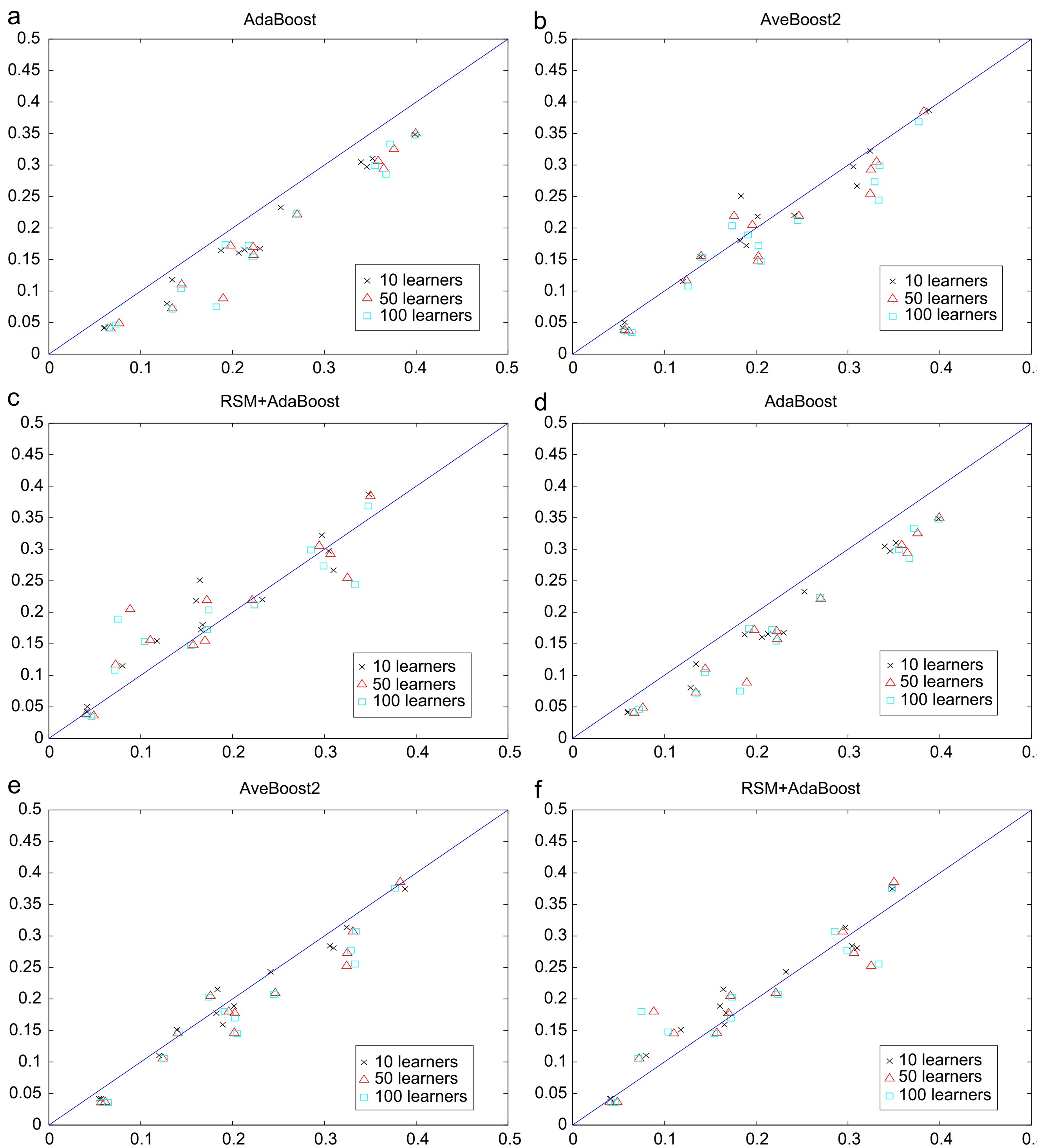

Fig. 3. Comparison of testing error between two algorithms based on k-NN component classifier. (a) AdaBoost (x-axis) and k-NN+ND (y-axis). (b) AveBoost2 (x-axis) and k-NN+ND (y-axis). (c) RSM+AdaBoost (x-axis) and k-NN+ND (y-axis). (d) AdaBoost (x-axis) and EM+ND (y-axis). (e) AveBoost2 (x-axis) and EM+ND (y-axis). (f) RSM+AdaBoost (x-axis) and EM+ND (y-axis).

the other hand, ND-AdaBoost has a good behavior with respect to testing performance even in different noise environment, especially at 5% noise level. As for Gentle AdaBoost, it performs better than AdaBoost and Modest AdaBoost although the differences are not significant.

Table 2 also gives the average testing error of each individual dataset. Without noise, the performances of all methods are roughly equal, but for 5% and 10% noise cases, ND-AdaBoost method achieves the best performance on 7 datasets and other methods only exhibit best behavior on 1 or 2 datasets. More specifically, ND-AdaBoost has the lowest testing errors on Breast, German, Heart and Ionosphere sets for all noise levels, and Real AdaBoost outperforms the others on Wdbc dataset.

5.2. Performance comparison of ND-AdaBoost with different base classifiers

In this study, we test our methodology against the selected boosting algorithms: Discrete AdaBoost, RSM AdaBoost and AveBoost2 [21] at high noise levels 10%, 20% and 30%. Since the

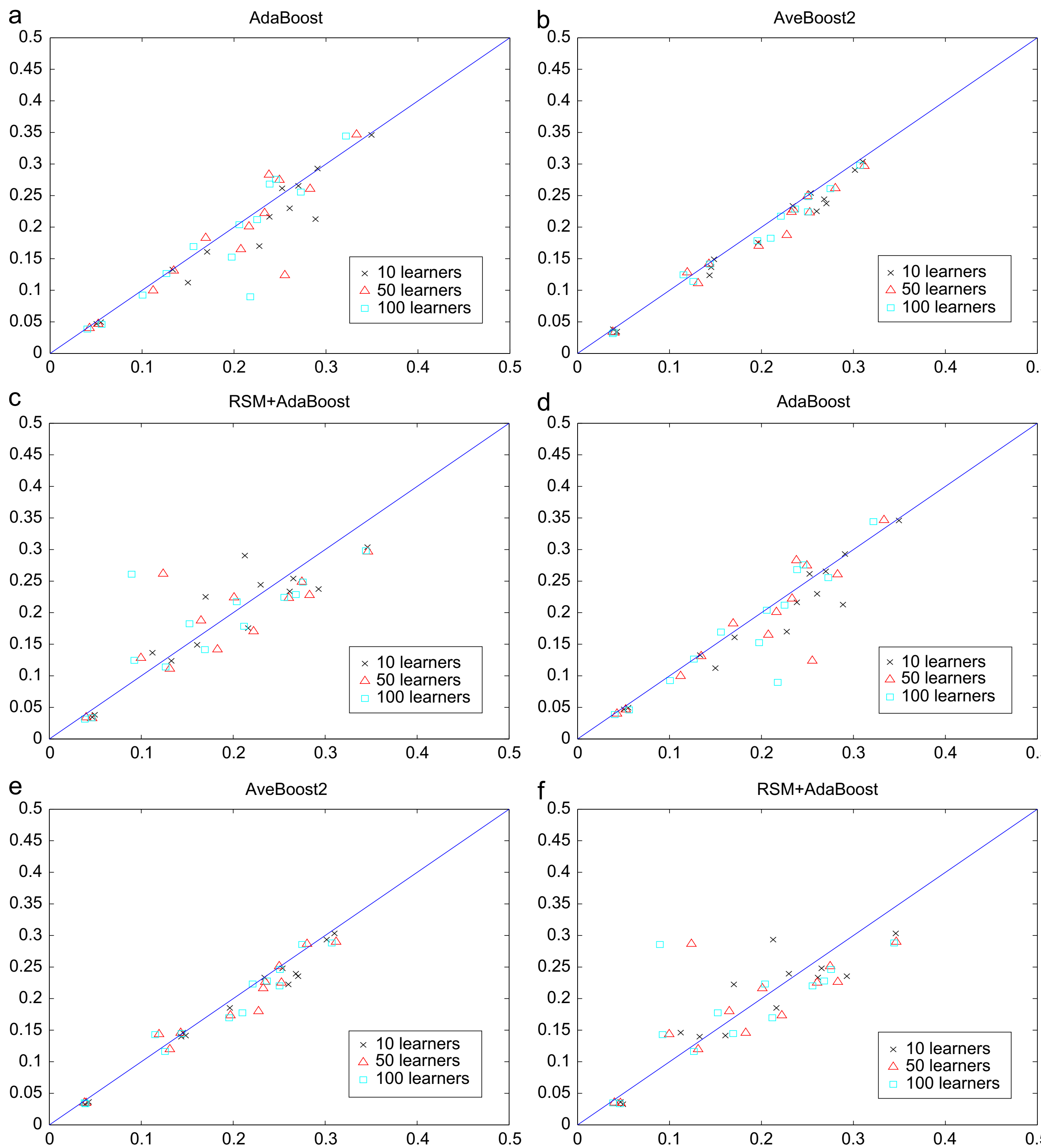

Fig. 4. Comparison of testing error between two algorithms based on ELM component classifier. (a) AdaBoost (*x*-axis) and *k*-NN+ND (*y*-axis). (b) AveBoost2 (*x*-axis) and *k*-NN+ND (*y*-axis). (c) RSM+AdaBoost (*x*-axis) and *k*-NN+ND (*y*-axis). (d) AdaBoost (*x*-axis) and EM+ND (*y*-axis). (e) AveBoost2 (*x*-axis) and EM+ND (*y*-axis). (f) RSM+AdaBoost (*x*-axis) and EM+ND (*y*-axis).

choice of different base classifiers is able to affect the performance of algorithm, we implement the experimental comparisons on three types of weak learners. As Section 4.3 stated, we propose two different functions: k-NN and EM. Both criteria are taken into account to test the noise tolerant. A 5×2 cross-validation (CV) is also conducted by repeatedly running two-fold cross-validation five times. The ensemble size is fixed at $T=100$.

5.2.1. Standard comparison

Figs. 3–5 represent the comparisons of classification error on test sets without noise impact by using k-NN, ELM and SVM as base classifiers. The maximum number of boosting iterations is set as 10, 50 and 100, respectively. In the figures, the x coordinate of each point is the testing error of other boosting algorithms, while the y coordinate of each point is the testing error of the

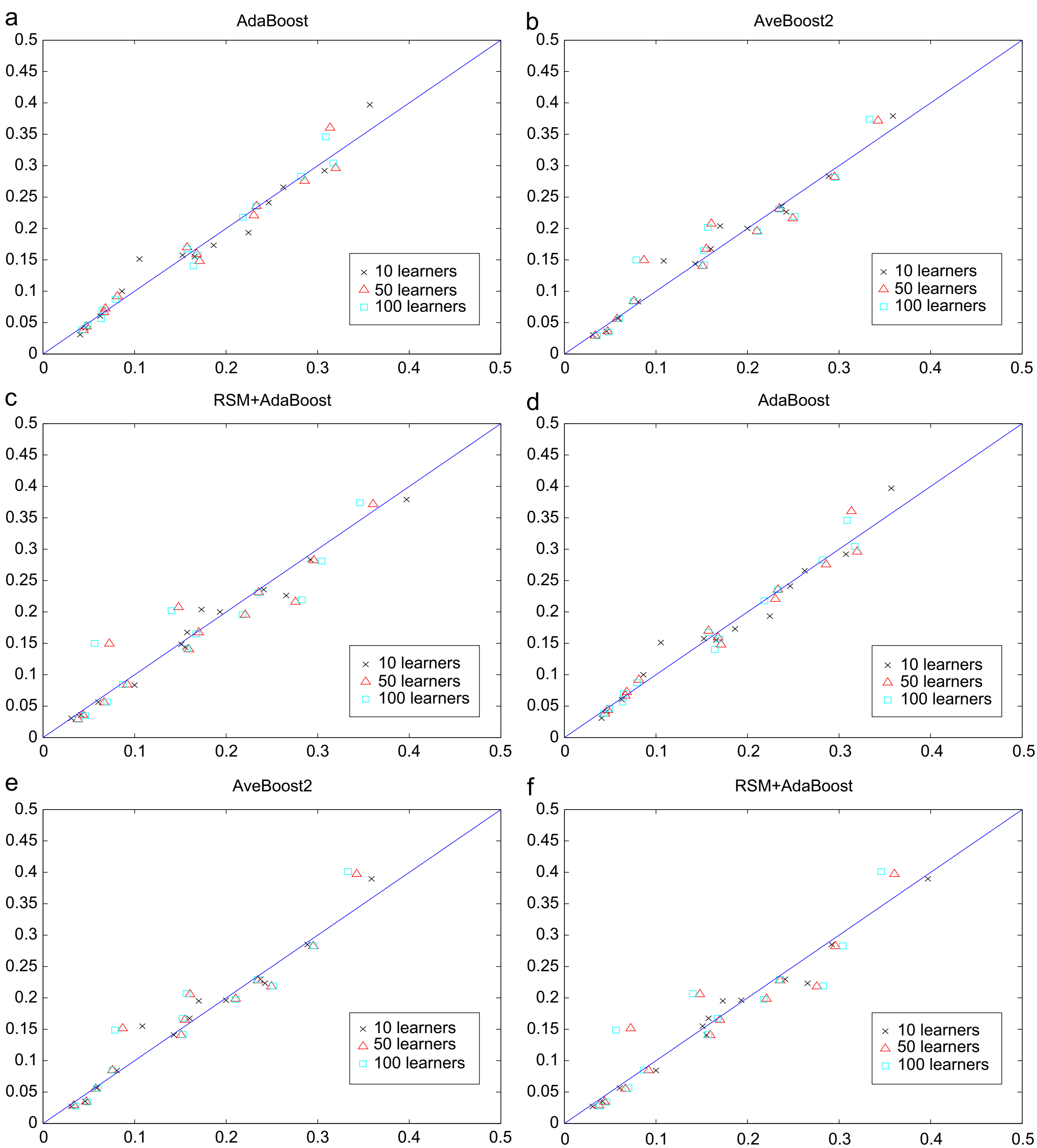

Fig. 5. Comparison of testing error between two algorithms based on SVM component classifier. (a) AdaBoost (x-axis) and k-NN+ND (y-axis). (b) AveBoost2 (x-axis) and k-NN+ND (y-axis). (c) RSM+AdaBoost (x-axis) and k-NN+ND (y-axis). (d) AdaBoost (x-axis) and EM+ND (y-axis). (e) AveBoost2 (x-axis) and EM+ND (y-axis). (f) RSM+AdaBoost (x-axis) and EM+ND (y-axis).

proposed algorithm with k-NN and EM based criteria, respectively. The more points that are below the diagonal line, the better performance our approach has. Considering k-NN classifier, from Fig. 3 we can see that both k-NN and EM based ND-AdaBoost perform worse than RSM AdaBoost, but they can outperform AdaBoost for all datasets, and AveBoost2 algorithm in most cases. In Fig. 4, the proposed methods have better behavior than AdaBoost and AveBoost2 for most datasets and are competitive with RSM AdaBoost by using ELM as component classifier. Regarding SVM, since it is a stable base classifier, boosting algorithms might not help improve testing error much. Thus, the performances for all the testing methods are nearly the same. In a summary, ND-AdaBoost algorithms are able to perform better than AdaBoost, AveBoost2 and RSM AdaBoost for most of datasets, but the differences are not significant.

5.2.2. Noise effect

The statistical significant comparisons of testing error with respect to different noise rates and weak learners are illustrated in Tables 3–11. Regarding statistical tests, sign test and Wilcoxon signed-rank test are employed to measure pairwise differences [38]. These tests are fairly popular in the literature [22,23,39,40]. For all the tables, "✓" is used if two methods are significant different with 95% confidence, while "•" is used with 90% confidence. Wins/ties/losses is also given as an index to compare the performances between pairwise algorithms. As for multiple comparisons, an Iman–Davenport test [41] is used to detect that whether there are significant statistical differences among all the algorithms.

From Tables 3–11, all tables state the remarkable behavior of the proposed method regardless of different noise environments and weak learners. In these tables, the p-values of sign test and Wilcoxon signed-rank test are denoted as p_s and p_w, respectively, and $w/t/l$ represents the wins/ties/losses record of each pair of classifiers. More specifically, for k-NN method, the performance of Discrete AdaBoost is poor, which empirically proves that AdaBoost does overfit. RSM AdaBoost algorithm shows some noise

Table 3
Statistical significant comparison of ensemble methods at noise rate $r=10\%$ with k-NN as base classifier.

Methods		AdaBoost	RSM	AveBoost2	ND+EM	ND+KNN
Mean		0.3200	0.2607	0.2918	0.2481	0.2474
AdaBoost	$w/t/l$		13/0/0	13/0/0	12/0/1	12/0/1
	p_s		0.0002✓	0.0002✓	0.0034✓	0.0034✓
	p_w		0.0002✓	0.0002✓	0.0005✓	0.0005✓
RSM	$w/t/l$			0/0/13	8/0/5	9/0/4
	p_s			0.0002✓	0.5811	0.2668
	p_w			0.0002✓	0.2439	0.2439
AveBoost2	$w/t/l$				12/0/1	12/0/1
	p_s				0.0034✓	0.0034✓
	p_w				0.0017✓	0.0012✓
ND+EM	$w/t/l$					6/1/6
	p_s					1.0000
	p_w					0.6221

Table 4
Statistical significant comparison of ensemble methods at noise rate $r=20\%$ with k-NN as base classifier.

Methods		AdaBoost	RSM	AveBoost2	ND+EM	ND+KNN
Mean		0.3808	0.3386	0.3516	0.3132	0.3115
AdaBoost	$w/t/l$		13/0/0	13/0/0	13/0/0	13/0/0
	p_s		0.0002✓	0.0002✓	0.0002✓	0.0002✓
	p_w		0.0002✓	0.0002✓	0.0002✓	0.0002✓
RSM	$w/t/l$			4/0/9	11/0/2	10/0/3
	p_s			0.2668	0.0225✓	0.0923•
	p_w			0.0266✓	0.0081✓	0.0266✓
AveBoost2	$w/t/l$				12/0/1	12/0/1
	p_s				0.0034✓	0.0034✓
	p_w				0.0005✓	0.0005✓
ND+EM	$w/t/l$					8/0/5
	p_s					0.5811
	p_w					0.3757

Table 5
Statistical significant comparison of ensemble methods at noise rate $r=30\%$ with k-NN as base classifier.

Methods		AdaBoost	RSM	AveBoost2	ND+EM	ND+KNN
Mean		0.4333	0.4011	0.3985	0.3743	0.3658
AdaBoost	$w/t/l$		12/0/1	13/0/0	13/0/0	13/0/0
	p_s		0.0034✓	0.0002✓	0.0002✓	0.0002✓
	p_w		0.0005✓	0.0002✓	0.0002✓	0.0002✓
RSM	$w/t/l$			7/0/6	12/0/1	11/0/2
	p_s			1.0000	0.0034✓	0.0225✓
	p_w			0.7354	0.0024✓	0.0034✓
AveBoost2	$w/t/l$				13/0/0	13/0/0
	p_s				0.0002✓	0.0002✓
	p_w				0.0002✓	0.0002✓
ND+EM	$w/t/l$					10/0/3
	p_s					0.0923•
	p_w					0.0803•

Table 6
Statistical significant comparison of ensemble methods at noise rate $r=10\%$ with ELM as base classifier.

Methods		AdaBoost	RSM	AveBoost2	ND+EM	ND+KNN
Mean		0.2414	0.2513	0.2367	0.2231	0.2175
AdaBoost	$w/t/l$		4/0/9	8/1/4	10/0/3	10/0/3
	p_s		0.2668	0.3877	0.0923•	0.0923•
	p_w		0.3396	0.4238	0.0574•	0.0061✓
RSM	$w/t/l$			10/0/3	10/0/3	11/0/2
	p_s			0.0923•	0.0923•	0.0225✓
	p_w			0.1099	0.0479✓	0.0105✓
AveBoost2	$w/t/l$				11/0/2	13/0/0
	p_s				0.0225✓	0.0002✓
	p_w				0.0081✓	0.0002✓
ND+EM	$w/t/l$					7/0/6
	p_s					1.0000
	p_w					0.4548

Table 7
Statistical significant comparison of ensemble methods at noise rate $r=20\%$ with ELM as base classifier.

Methods		AdaBoost	RSM	AveBoost2	ND+EM	ND+KNN
Mean		0.2974	0.3093	0.2969	0.2716	0.2712
AdaBoost	$w/t/l$		3/0/10	6/0/7	11/0/2	12/0/1
	p_s		0.0923	1.0000	0.0225✓	0.0034✓
	p_w		0.1125	0.9460	0.0061✓	0.0005✓
RSM	$w/t/l$			9/0/4	10/0/3	10/0/3
	p_s			0.2668	0.0923•	0.0923•
	p_w			0.2734	0.0081✓	0.0061✓
AveBoost2	$w/t/l$				12/0/1	13/0/0
	p_s				0.0034✓	0.0002✓
	p_w				0.0007✓	0.0002✓
ND+EM	$w/t/l$					5/0/8
	p_s					0.5811
	p_w					0.5879

Table 8
Statistical significant comparison of ensemble methods at noise rate $r=30\%$ with ELM as base classifier.

Methods		AdaBoost	RSM	AveBoost2	ND+EM	ND+KNN
Mean		0.3501	0.3696	0.3425	0.3251	0.3211
AdaBoost	$w/t/l$		4/0/9	8/0/5	10/0/3	10/0/3
	p_s		0.2668	0.5811	0.0923•	0.0002✓
	p_w		0.0681	0.1465	0.0034✓	0.0002✓
RSM	$w/t/l$			13/0/0	10/0/3	11/0/2
	p_s			0.0923•	0.0923•	0.0225✓
	p_w			0.0171✓	0.0034✓	0.0017✓
AveBoost2	$w/t/l$				11/0/2	13/0/0
	p_s				0.0225✓	0.0002✓
	p_w				0.0017✓	0.0002✓
ND+EM	$w/t/l$					6/0/7
	p_s					1.0000
	p_w					0.5417

Table 9
Statistical significant comparison of ensemble methods at noise rate $r=10\%$ with SVM as base classifier.

Methods		AdaBoost	RSM	AveBoost2	ND+EM	ND+KNN
Mean		0.2345	0.2322	0.2165	0.2087	0.2064
AdaBoost	$w/t/l$		8/0/5	12/0/1	8/0/5	11/0/2
	p_s		0.5811	0.0034✓	0.5811	0.0225✓
	p_w		0.3757	0.0081✓	0.0681•	0.0081✓
RSM	$w/t/l$			12/0/1	10/0/3	11/0/2
	p_s			0.0034✓	0.0923•	0.0225✓
	p_w			0.0017✓	0.0105✓	0.0034✓
AveBoost2	$w/t/l$				8/0/5	10/0/3
	p_s				0.5811	0.0923•
	p_w				0.1465	0.0215✓
ND+EM	$w/t/l$					7/1/5
	p_s					0.7744
	p_w					0.2661

Table 10
Statistical significant comparison of ensemble methods at noise rate $r=20\%$ with SVM as base classifier.

Methods		AdaBoost	RSM	AveBoost2	ND+EM	ND+KNN
Mean		0.2985	0.2946	0.2735	0.2598	0.2576
AdaBoost	$w/t/l$		8/0/5	11/1/1	11/0/2	12/0/1
	p_s		0.5811	0.0063✓	0.0225✓	0.0034✓
	p_w		0.5417	0.0024✓	0.0034✓	0.0007✓
RSM	$w/t/l$			13/0/0	12/0/1	12/0/1
	p_s			0.0002✓	0.0034✓	0.0034✓
	p_w			0.0002✓	0.0007✓	0.0005✓
AveBoost2	$w/t/l$				11/0/2	12/0/1
	p_s				0.0225✓	0.0034✓
	p_w				0.0061✓	0.0012✓
ND+EM	$w/t/l$					8/1/4
	p_s					0.3877
	p_w					0.2036

resistance. Though it can significantly outperform AveBoost2, it has no statistical difference with the proposed methods at 10% noise rate. Nevertheless, with more noisy instances being added, its performances are gradually similar as AveBoost2 but both methods fail to compete with the proposed ND-AdaBoost method. Further, it is interesting to note that at 30% noise level, the k-NN based ND-AdaBoost is statistically superior to the EM based ND-AdaBoost with 90% confidence. Regarding ELM, AdaBoost, AveBoost2 and RSM AdaBoost show no significant difference below 30% noise rate. Except that AveBoost2 performs significantly better than RSM AdaBoost at 30% noise level. Considering using SVM as weak learner, AveBoost2

Table 11
Statistical significant comparison of ensemble methods at noise rate $r=30\%$ with SVM as base classifier.

Methods		AdaBoost	RSM	AveBoost2	ND+EM	ND+KNN
Mean		0.3640	0.3583	0.3418	0.3255	0.3251
AdaBoost	$w/t/l$		9/0/4	10/0/3	10/0/3	12/0/1
	p_s		0.2668	0.0923•	0.0923•	0.0034✓
	p_w		0.2734	0.0046✓	0.0134✓	0.0007✓
RSM	$w/t/l$			11/0/2	11/0/2	12/0/1
	p_s			0.0225✓	0.0225✓	0.0034✓
	p_w			0.0012✓	0.0034✓	0.0007✓
AveBoost2	$w/t/l$				9/0/4	11/0/2
	p_s				0.2668	0.0225✓
	p_w				0.0327✓	0.0034✓
ND+EM	$w/t/l$					8/0/5
	p_s					0.5811
	p_w					0.6848

performs much better than both RSM AdaBoost and AadBoost, but still much worse than ND-AdaBoost. Regarding the p-values returned by the Iman–Davenport test, all are smaller than 0.01 and reject the null hypothesis that all algorithms are equivalent. In detail, the Iman–Davenport test obtaining p-values of 0.0000 at any noise level using k-NN. For ELM and SVM, the returned p-values are 0.0002, 0.0000, 0.0000 and 0.0004, 0.0000, 0.0000 at noise levels 10%, 20% and 30%, respectively.

5.3. Margin distribution

The margin comparison on training instances is an effective plot to illustrate the performance of boosting algorithm. The margin distribution graphs [31] show the proportion of instance whose margin is at most x as a function of $x \in [-1, 1]$. Figs. 6 and 7 show the margin distribution results after 100 iterations for all the 13 data sets at 10% noise level using SVM as weak learner. We can see that the AdaBoost and RSM AdaBoost algorithms tend to concentrate on training classifiers with large margins on positive instances. Note that an instance with positive margin is considered to be correctly classified. The training errors of both algorithms are zero and this phenomenon leads to the over-fitting problem. In contrast, our proposed algorithm tries to identify noisy instances and non-noisy instances, and focuses on making noisy instances misclassified while non-noisy instances correctly classified. Thus, we allow a large negative margin associated with a proportion of instances. AveBoost2 shows a moderate negative margin among all methods. As far as we know, one type of measurement to evaluate generalization ability of classifier is to calculate the difference between the training and testing errors [27]. The less difference indicates the higher generalization ability. Thus, from Table 9 and margin plots (Figs. 6 and 7), it can be seen that the proposed k-NN and EM based algorithms show better generalization ability, since compared with the other three methods, a better testing performance and a worse training performance lead to a smaller difference between testing and training errors. Furthermore, the proposed classifiers are more stable due to the fact that most misclassified instances are with large negative margins, whose labels are unlikely to be changed by small changes of combination weights.

As a conclusion, we can see that k-NN and EM based ND-AdaBoost can significantly outperform than Discrete AdaBoost, RSM AdaBoost and AveBoost2 in the presence of noise.

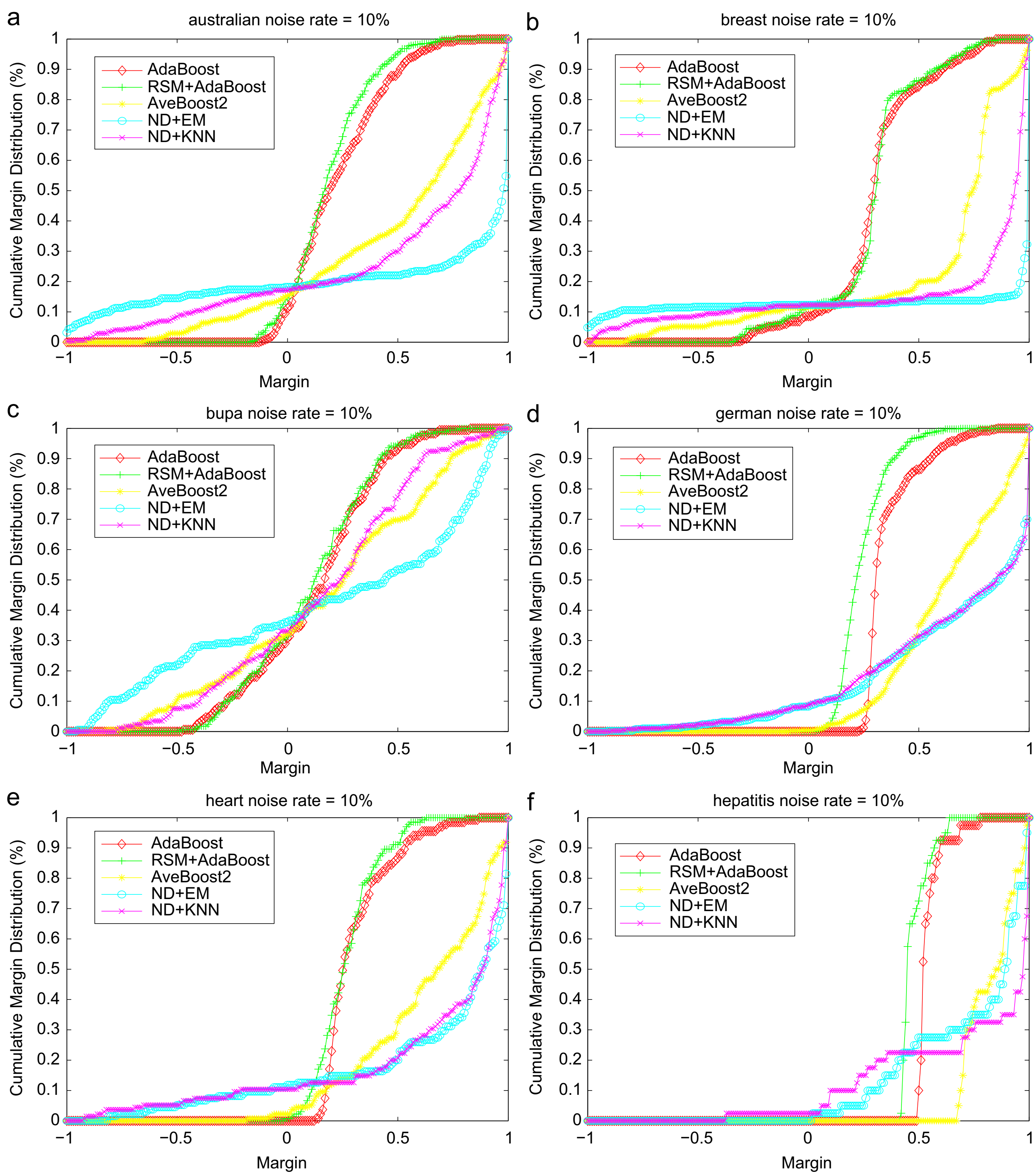

Fig. 6. Margin distributions for AadBoost, RSM AdaBoost, AveBoost2, ND+EM and ND+KNN. (a) Australian, (b) Breast, (c) Bupa, (d) German, (e) Heart, (f) Hepatitis data set.

6. Conclusion

To solve the incompatibility between AdaBoost algorithm and noisy instances, this paper designs a noise-detection version of boosting algorithm. It labels the noisy instances at each iteration and adds a regeneration condition to control the ensemble training error bound. The experimental results with artificially added mislabeled instances show that different noise identification functions (k-NN and EM based) affect the performance of ND-AdaBoost algorithm, i.e., the testing result of k-NN based method is slightly better than EM based method. Both of them significantly outperform most of the state-of-the-art methods under high noisy environments but show some improvements at low noisy levels. Finally, it can be concluded

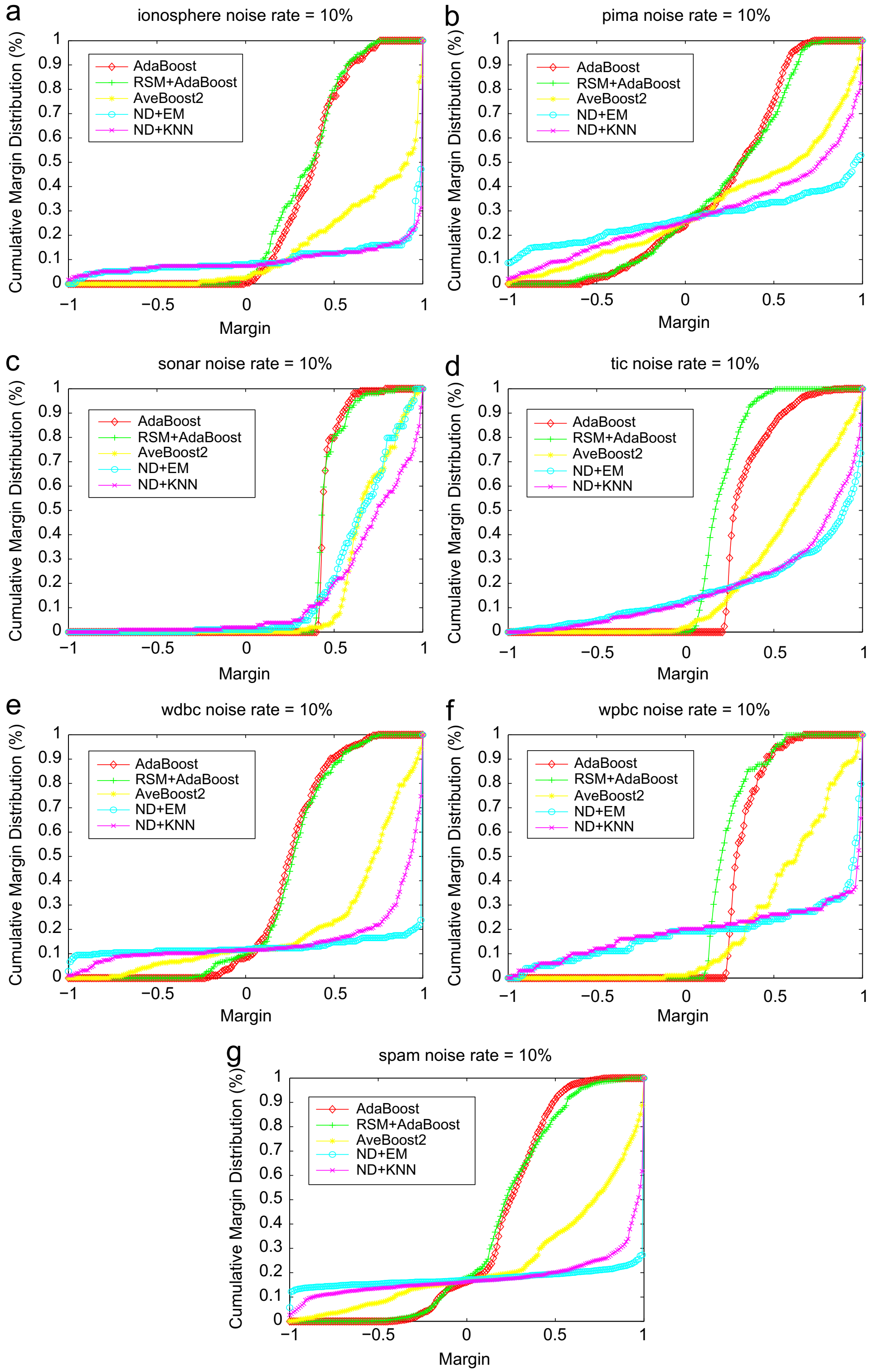

Fig. 7. Margin distributions for AadBoost, RSM AdaBoost, AveBoost2, ND+EM and ND+KNN. (a) Ionosphere, (b) Pima, (c) Sonar, (d) Tic, (e) Wdbc, (f) Wpbc, (g) Spam data set.

that the proposed algorithm is stable and insensitive to the noise-detection function selection.

Acknowledgements

This work is supported by City University Grant 9610025 and City University Strategic Grant 7002680. The authors would like to acknowledge the two anonymous reviewers' comments, which contributed to improve the quality of our paper significantly.

References

[1] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: ICML, 1996, pp. 148–156.
[2] J. Quinlan, Bagging, boosting, and C4. 5, in: Proceedings of the National Conference on Artificial Intelligence, 1996, pp. 725–730.
[3] F. Yoav, E. Robert, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1) (1997) 119–139.
[4] Y. Sun, M. Kamel, A. Wong, Y. Wang, Cost-sensitive boosting for classification of imbalanced data, Pattern Recognition 40 (12) (2007) 3358–3378.

[5] H. Masnadi-Shirazi, N. Vasconcelos, Cost-sensitive boosting, IEEE Transactions on Pattern Analysis and Machine Intelligence (2010) 294–309.
[6] P.K. Mallapragada, R. Jin, A.K. Jain, Y. Liu, SemiBoost: boosting for semi-supervised learning, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (11) (2009) 2000–2014.
[7] S. Avidan, Ensemble tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence (2007) 261–271.
[8] W. Hu, W. Hu, S. Maybank, Adaboost-based algorithm for network intrusion detection, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 38 (2) (2008) 577–583.
[9] R. Schapire, The strength of weak learnability, Machine Learning 5 (2) (1990) 197–227. ISSN 0885-6125.
[10] G. Rätsch, T. Onoda, K.-R. Müller, Soft margins for adaBoost, Machine Learning 42 (3) (2001) 287–320.
[11] T. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, Machine Learning 40 (2) (2000) 139–157. ISSN 0885-6125.
[12] C. Bishop, Pattern Recognition and Machine Learning, vol. 4, Springer, New York, 2006.
[13] A. Frank, A. Asuncion, UCI Machine Learning Repository 〈http://archive.ics.uci.edu/ml〉, 2010.
[14] J. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting, The Annals of Statistics 28 (2) (2000) 337–374.
[15] R. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Machine Learning 37 (3) (1999) 297–336.
[16] G. Lemaıtre, M. Radojevic, Directed Reading: Boosting algorithms. 〈http://g.lemaitre58.free.fr/pdf/vibot/english_writting/literature_review.pdf〉, 2009.
[17] A. Vezhnevets, V. Vezhnevets, Modest adaBoost-teaching adaBoost to generalize better, Graphicon-2005. Novosibirsk Akademgorodok, Russia, 2005.
[18] A. Ferreira, Survey on Boosting Algorithms for Supervised and Semi-supervised Learning, Technical Report, Instituto Superior de Engenharia de Lisboa, 2007.
[19] C. Domingo, O. Watanabe, MadaBoost: a modification of adaBoost, in: COLT, 2000, pp. 180–189.
[20] Y. Gao, F. Gao, X. Guan, Improved boosting algorithm with adaptive filtration, in: 2010 8th World Congress on Intelligent Control and Automation (WCICA), IEEE, 2010, pp. 3173–3178.
[21] N.C. Oza, AveBoost2: boosting for noisy data, in: Multiple Classifier Systems, 2004, pp. 31–40.
[22] N. García-Pedrajas, Supervised projection approach for boosting classifiers, Pattern Recognition 42 (9) (2009) 1742–1760.
[23] N. García-Pedrajas, C. García-Osorio, Constructing ensembles of classifiers using supervised projection methods based on misclassified instances, Expert Systems with Applications 38 (1) (2011) 343–359.
[24] Y. Gao, F. Gao, Edited adaBoost by weighted kNN, Neurocomputing 73 (16–18) (2010) 3079–3088.
[25] U. Rebbapragada, C. Brodley, Class noise mitigation through instance weighting, Machine Learning: ECML 2007 (2007) 708–715.
[26] T. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 832–844.
[27] S. Shirai, M. Kudo, A. Nakamura, Comparison of bagging and boosting algorithms on sample and feature weighting, Multiple Classifier Systems (2009) 22–31.
[28] N. García-Pedrajas, D. Ortiz-Boyer, Boosting random subspace method, Neural Networks 21 (9) (2008) 1344–1362.
[29] L. Nanni, A. Franco, Reduced reward-punishment editing for building ensembles of classifiers, Expert Systems with Applications 38 (3) (2011) 2395–2400.
[30] M. Collins, R. Schapire, Y. Singer, Logistic regression, adaBoost and Bregman distances, Machine Learning 48 (1) (2002) 253–285.
[31] R. Schapire, Y. Freund, P. Bartlett, W. Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, The Annals of Statistics 26 (5) (1998) 1651–1686. ISSN 0090-5364.
[32] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Academic Press, 2003.
[33] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1) (1967) 21–27.
[34] A. Vezhnevets, Gml adaboost matlab toolbox, Technique Manual, Graphics and Media Laboratory, Computer Science Department, Moscow State University, Moscow, Russian Federation.
[35] G. Huang, Q. Zhu, C. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[36] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, software available at 〈http://www.csie.ntu.edu.tw/〉 cjlin/libsvm, 2001.
[37] L. Breiman, Classification and Regression Trees, Chapman & Hall, CRC, 1984.
[38] J. Demšar, Statistical comparisons of classifiers over multiple data sets, The Journal of Machine Learning Research 7 (2006) 1–30.
[39] N. Garcia-Pedrajas, Constructing ensembles of classifiers by means of weighted instance selection, IEEE Transactions on Neural Networks 20 (2) (2009) 258–277. ISSN 1045-9227.
[40] M. Galar, A. Fernández, E.B. Tartas, H.B. Sola, F. Herrera, An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes, Pattern Recognition 44 (8) (2011) 1761–1776.
[41] R. Iman, J. Davenport, Approximations of the critical region of the Friedman statistic, Communications in Statistics—Theory and Methods 9 (6) (1980) 571–595.

Jingjing Cao received her B.S. degree in Information and Computing Science, Dalian Maritime University, China in 2006 and the M.S. degree in Applied Mathematics from the same university in 2008. She is currently doing Ph.D. in the Department of Computer Science at City University of Hong Kong, Kowloon, Hong Kong. Her research interests are in Ensemble Learning, Evolutionary Algorithms, and their applications.

Sam Kwong received his BSc degree and MASc degree in electrical engineering from the State University of New York at Buffalo, USA, and University of Waterloo, Canada, in 1983 and 1985, respectively. In 1996, he later obtained his PhD from the University of Hagen, Germany. From 1985 to 1987, he was a diagnostic engineer with the Control Data Canada where he designed the diagnostic software to detect the manufacture faults of the VLSI chips in the Cyber 430 machine. He later joined the Bell Northern Research Canada as a Member of Scientific staff. In 1990, he joined the City University of Hong Kong as a lecturer in the Department of Electronic Engineering. He is currently a Professor in the Department of Computer Science. His research interests are in Pattern Recognition, Evolutionary Algorithms and Video Coding.

Ran Wang received her Bachelor's degree from School of Information Science & Technology, Beijing Forestry University, Beijing, China, in 2009. She is currently a Ph.D candidate in the Department of Computer Science, City University of Hong Kong. Her research interests focus on machine learning and its related applications.