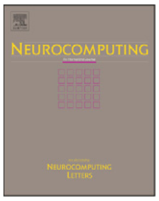
Neurocomputing 73 (2010) 3079–3088

Contents lists available at ScienceDirect

Neurocomputing

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



Edited AdaBoost by weighted kNN

Yunlong Gao*, Feng Gao

Systems Engineering Institute, Xi'an Jiaotong University, Xi'an 710049, China

A R T I C L E   I N F O

Article history:
Received 2 December 2009
Received in revised form
30 May 2010
Accepted 28 June 2010
Communicated by G.-B. Huang
Available online 25 August 2010

Keywords:
AdaBoost
Overfitting
kNN rules
Adaptability
Feature space

A B S T R A C T

Any realistic model of learning from samples must address the issue of noisy data. AdaBoost is known as an effective method for improving the performance of base classifiers both theoretically and empirically. However, previous studies have shown that AdaBoost is prone to overfitting, especially in noisy domains. On the other hand, the kNN rule is one of the oldest and simplest methods for pattern classification. Nevertheless, it often yields competitive results, and in certain domains, when cleverly combined with prior knowledge, it has significantly advanced the state-of-the-art. In this paper, an edited AdaBoost by weighted kNN (EAdaBoost) is designed where AdaBoost and kNN naturally complement each other. First, AdaBoost is run on the training data to capitalize on some statistical regularity in the data. Then, a weighted kNN algorithm is run on the feature space composed of classifiers produced by AdaBoost to achieve competitive results. AdaBoost is then used to enhance the classification accuracy and avoid overfitting by editing the data sets using the weighted kNN algorithm for improving the quality of training data. Experiments performed on ten different UCI data sets show that the new Boosting algorithm almost always achieves considerably better classification accuracy than AdaBoost. Furthermore, experiments on data with artificially controlled noise indicate that the new Boosting algorithm is robust to noise.

1. Introduction

Despite the fact that learning in the presence of noxious noise is generally quite difficult, the importance of being able to cope with noisy data has led many researchers to study PAC learning in the presence of malicious noise. AdaBoost, the most promising PAC learning algorithm, has been proved theoretically and shown empirically to be an effective method for improving the classification accuracy [1–5]. It was believed initially, that AdaBoost seldom overfits the training data. Even though the training error reaches zero, the AdaBoost algorithm still has a lower test error while training. However, recent studies have suggested that AdaBoost might suffer from the problem of overfitting [6–10], especially for noisy data sets.

The main advantage of AdaBoost over other boosting techniques is that it is adaptive, i.e., it is able to take advantage of weak hypotheses to maximize the minimum margin even if the training error of the combination of hypotheses is zero [10–12]. There are theoretical bounds on the generalization error of linear classifiers [11,13], which decrease as the smallest margin of the samples increases. Hence, the adaptive character of AdaBoost ensures that it creates hypotheses with good generalization. The adaptiveness of AdaBoost, however, is a double-edged sword, in the sense that for noisy data sets, there could be certain data samples that are difficult for the classifier to capture. The boosting algorithm then tends to concentrate its resources on these suspect samples [14], thereby distorting the optimal decision boundary. As a result, the decision boundary will only be suitable for those difficult data samples and not necessarily general enough for other data. The margin maximized by AdaBoost is actually a 'hard margin', that is, the smallest margin of the noisy data samples. Consequently, the margin of the other data points may decrease significantly when we maximize the 'hard margin', thereby forcing the generalized error bound to increase.

Clearly, the property of concentrating AdaBoost resources on a few suspect samples leads to overfitting. The most effective approach for dealing with the overfitting problem in AdaBoost is to eliminate the harmful effects of suspect samples. Several strategies have been proposed to cope with overfitting. The key ideas of these methods can be summarized into two groups: the first attempts to reduce the effects of some hard-to-learn samples, while the other eliminates the effects of hard-to-learn samples. However, to avoid overfitting, two crucial problems should be noted. Which hard-to-learn samples can be dealt with as suspect samples? And, how many hard-to-learn samples can be dealt with as suspect samples?

The rest of the paper is arranged as follows. In Section 2, we discuss related work on other boosting algorithms. In Section 3, we present a weighted kNN algorithm for distinguishing samples into two categories: suspect and non-suspect samples. Then, a full description of EAdaBoost is presented in Section 4; some analyses

* Corresponding author. Tel.: +86 13289368676.
E-mail addresses: ylgao@sei.xjtu.edu.cn, gaoyl05@yahoo.com.cn (Y. Gao).

0925-2312/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.06.024

of our proposed algorithm are performed too in Section 4. Finally, Section 5 describes our experiments, while Section 6 concludes the paper.

2. Related work

In most works, estimating the 'hardness' of every training example is based on observations of algorithm behavior. The harmful effects of some hard-to-learn samples are reduced by a modification weighting scheme or by introducing a different cost function. As an example of this, Servedio [15] proposed a smooth boosting algorithm using a modified weighting scheme. This algorithm generates only smooth distributions that do not assign too much weight to any single sample. Ratsch et al. [16] analyzed the dynamic evolution of AdaBoost weights for estimating the 'hardness' of every training sample and regularized the exponential cost function with a penalty term, such as the weight decay method. Based on the concept of robust statistics, Kanamori et al. [17] proposed a transformation of loss functions that makes boosting algorithms robust against extreme outliers.

All the works described above can be characterized as reducing the harmful effects of hard-to-learn samples to avoid overfitting, and implicitly solve the problems of which and how many samples can be dealt with as suspect samples by weight shrinking. Although these weight shrinking methods seem to be very promising, researchers have not reported any significant difference between AdaBoost and the methods of weight shrinking in experiments on noisy data. For example, Domingo and Watanabe [18] proposed a modification of AdaBoost in which the weights of the samples are kept bounded by its initial value. Nevertheless, the authors have reported no significant difference between AdaBoost and this modification in experiments on noisy data. Vezhnevets and Barinova [19] showed that, despite regularization, MadaBoost is also prone to overfitting as the algorithm approaches termination.

A large body of research demonstrates that removing hard samples is worthwhile [20–22]. The main goal of these approaches is to enhance the classification accuracy by eliminating the harmful effects of suspect samples. As the algorithm approaches its predetermined end, it is less and less likely that samples with large negative margins will eventually be correctly labeled. Thus, it is more beneficial for the algorithms to 'give up' on these samples and concentrate their efforts on those samples with small negative margins. Examples of such algorithms are *BrownBoost* and regularized boosting algorithms, which eliminate the harmful effects of suspect samples by 'giving up' on these samples. These methods estimate the degree of 'hardness' of the training sample based on the influence of the sample on the combined hypotheses. However, they determine how many samples can be dealt with as suspect samples based on certain important parameters fixed in advance. For example, to use *BrownBoost* one needs to pre-specify an upper bound $(1/2)-\gamma$ on the error of a weak learner and a 'target' error $\varepsilon > 0$ should be given a priori [7]. Regularization methods [23] need to fix a regularization constant in advance.

The problem of handling mislabeled, atypical and noisy training samples has been the focus of much attention in both pattern recognition and machine learning domains. The k-nearest neighbors (kNN) rule is one of the oldest, simplest and non-parametric methods for improving the quality of the training data [24]. However, the performance of kNN depends crucially on the distance metric used to identify nearest neighbors. In fact, as shown by many researchers [25–28], kNN classification can be significantly improved if the input features can capitalize on any statistical regularities in the data. Even a simple (global) linear transformation of input features has been shown to yield much better kNN classifiers.

Since the feature space composed of the obtained classifier in AdaBoost capitalizes on some statistical regularities in the data, in this paper, we first give a weighted kNN algorithm which takes full advantage of the feature space. Then, we propose a new boosting method called Edited AdaBoost by weighted kNN (EAdaBoost). In each iteration of the new method, the weighted kNN algorithm is used to distinguish samples into two categories: suspect and non-suspect samples. A suspicious sample is a sample, for which the optimal prediction for a given loss, and the family of classifiers, differs from its current label, or those observations regarding rare samples, which are often associated with variable asymmetry and are an important cause for leverage points [29]. Samples viewed as being suspicious should have their influence eliminated in the next iteration. In each cycle of the new algorithm, the quality of the training data is improved by 'giving up' on some suspect samples, thus ensuring that the new algorithm enhances classification accuracy and avoids overfitting. In contrast to the works described above which try to cope with overfitting in AdaBoost, the new method explicitly and strictly defines which and how many samples can be dealt with as suspect samples in each cycle. Moreover, none of the important additional parameters need to be given in advance in the new algorithm.

3. Weighted kNN

One of the most widely studied non-parametric classification approaches corresponds to the kNN rule [24,30]. The goal of the kNN algorithm is to form a generalization from a set of labeled training samples such that the classification accuracy for new samples is maximized. The maximum accuracy achievable depends on the quality of the input data and on the appropriateness of the chosen k nearest neighbors. In this section, we give a brief description of the kNN algorithm. After describing the statistical basis of the kNN method, we present a new weighted kNN algorithm, which focuses mainly on how to choose the k nearest neighbors.

3.1. KNN classification rule

For convenience, we fix some terminology. Let $S=(x_i, y_i)$, $i=1,2,\ldots,N$ be the training set, where x_i is a $d-$dimensional vector of attributes and $y_i \in \{+1,-1\}$ is the associated observed class label (for simplicity, we consider a binary classification task). To justify generalization, we make the assumption that the training data are *iid* samples of random variables (X, Y) having some unknown distribution.

Given N previously labeled samples as the training set S, the kNN algorithm constructs a local subregion $R(x) \subseteq \mathbb{R}^d$ of the input space, centered at the estimation point x. The predicting region $R(x)$ contains the k closest training points to x:

$$R(x) = \{\hat{x} | D(x,\hat{x}) \le d_{(k)}\} \tag{1}$$

where $d_{(k)}$ is the kth order statistic of $\{D(x,\hat{x})\}_1^N$, and $D(x,\hat{x})$ is a distance metric. Let $k[y]$ denote the number of samples in region $R(x)$, which are labeled y. The kNN algorithm is statistically inspired in the estimation of the posterior probability $P(y|x)$ of the observation point x:

$$p(y|x) = \frac{p(y|x)p(y)}{p(x)} \cong \frac{k[y]}{k} \tag{2}$$

For a given observation x, the decision $g(x)$ is taken by evaluating the values of $k[y]$, and selecting the class that has the highest $k[y]$ value:

$$g(x)=\begin{cases}1 & \text{if } k[y=1]\geq k[y=-1]\\ -1 & \text{else } k[y=1]<k[y=-1]\end{cases} \quad (3)$$

Thus, the decision that maximizes the associated posterior probability is taken in the kNN [31,32]. For a binary classification problem in which $y_i \in \{+1,-1\}$, then the kNN rule amounts to the following decision rule:

$$g(x)=sgn(ave_{x_i\in R(x)}y_i) \quad (4)$$

3.2. Weighted kNN algorithm

If we have infinitely many sample points, the estimation of the posterior probability converges to the actual posterior probability density function, and the kNN classifier becomes a Bayesian classifier. Accurate probability estimates can lead to better classification performance in the kNN. In practice, however, it is impossible to give infinitely many sample points. To reduce the error rate of the kNN, how to define the proximity between samples in the representation space becomes a critical issue.

A good idea is to choose k samples which are the most similar to a given observation x. Usually, the l_q distance is used to measure the k nearest neighbors, because the samples gather together by classes, so it is natural to assume that the interest belongs to the class of its nearest neighbors. Euclidean distance (l_q distance with $q=2$) is one of the methods used for measuring the similarities between samples in the representation space. The problem is that this distance metric does not consider the feature relevance to solve the classification task and so all feature distances contribute equally to select the k closest cases.

A promising idea to enhance the performance of a kNN algorithm is to find a new representation space [27,33,34], where each feature is weighted according to its ability for maximizing some discriminability criteria in the input data. In this space, we have designed a weighted kNN algorithm, which is described in Table 1.

Clearly, the only difference between the weighted kNN algorithm and the kNN is that we map labeled data samples into a new representation space. In the new representation space, each feature is weighted according to its ability to maximize some discriminability criteria in the input data. In addition, the kNN algorithm is executed in this new space. Now the key questions are how to build the new representation space, and to learn a distance metric used to identify nearest neighbors in the new space.

Table 1
A weighted kNN algorithm (S, ψ, Λ, k).

(1) Let ψ denote a new J-dimensional representation space with features placed as columns, and Λ the diagonal matrix with only nonzero values in the diagonal. A nonzero value in Λ denotes the weight of a feature calculated by its ability to maximize some discriminability criteria in the input data
(2) Mapping the estimation point x into the new space is denoted by x', where $x'=\Lambda\psi^T x$. Mapping any other point in the training set into the new space is denoted by $\tilde{x}$
(3) Construct a local subregion $R(x')\subseteq \mathbb{R}^J$ of the new space. $R(x')=\{\tilde{x}\,|\,D(x',\tilde{x})\leq d_{(k)}\}$, where $d_{(k)}$ is the kth order statistic of $\{D(x',\tilde{x})\}_1^N$ and $D(x',\tilde{x})$ is a distance metric used to identify nearest neighbors in the new space
(4) Calculate the decision $g(x)$ for a given observation x $g(x)=sgn(ave_{\tilde{x}_i\in R(x')}y_i)$ where $\tilde{x}_i=\Lambda\psi^T x_i$ and $x_i\in S$

4. AdaBoost and a connection with the weighted kNN algorithm

AdaBoost is a well-known PAC-model algorithm that usually calls a given learning algorithm repeatedly, trying to build a strong classifier that combines weak hypotheses to achieve more separability of the entire training data set. Then a representation space is defined by the hypothesis set. Each hypothesis defines one dimension in this space, with the weight of a hypothesis calculated by evaluating its classification error. It is natural to build a new representation space for the weighted kNN algorithm by using the AdaBoost algorithm.

In this section, we focus on learning a distance metric used to identify nearest neighbors in the new space. Then EAdaBoost (edited Adaboost) is presented, combining AdaBoost with the weighted kNN algorithm to avoid overfitting.

4.1. Distance metric in feature space

In AdaBoost, after J iterations, we construct a classifier $f(x)$ by the linear combination of the base classifier $h(x)$,

$$f(x)=\sum_{t=1}^{J}a_t h_t(x) \quad (5)$$

where $a_t(a_t>0)$ is the linear combination coefficient for the tth basic classifier $h_t(x)$, $h_t(x)\in H$, and H is the hypothesis space. The l_1-margin of sample x_i is $\rho(x_i,y_i)=(y_i f(x_i))/(\|\hat{a}\|_1)=(y_i\sum_{t=1}^{J}a_t h_t(x_i))/(\|\hat{a}\|_1)$, where $\hat{a}=[a_1,a_2,\ldots,a_J]^T$ and $\|\hat{a}\|_1$ denotes the l_1-norm. A positive margin corresponds to a correct classification, with the more positive the margin, the greater the confidence that the classification is correct.

The margin can be understood in the feature space, which is defined by the hypothesis set. Each hypothesis from H defines one dimension in this space. After J iterations, the hypothesis set is finite and has J elements. Samples are mapped into the feature space,

$$\tilde{x}=\psi(x)=[h_1(x),h_2(x),\ldots,h_J(x)]^T \quad (6)$$

and the l_1-margin of sample x_i in the feature space is

$$\rho(x_i,y_i)=\frac{y_i\langle\tilde{x}_i,\hat{a}\rangle}{\|\hat{a}\|_1} \quad (7)$$

where $\tilde{x}_i=\psi(x_i)$. According to (7), the distance between the decision boundary and the sample is $(|\langle\tilde{x}_i,\hat{a}\rangle|)/(\|\hat{a}\|_1)$.

Based on the definition of distance between the decision boundary and sample, we define the distance between two samples x_i and x_j as

$$D(x_i,x_j)=\frac{|\langle\tilde{x}_i-\tilde{x}_j,\hat{a}\rangle|}{\|\hat{a}\|_1}=\frac{|\langle\tilde{x}_i,\hat{a}\rangle-\langle\tilde{x}_j,\hat{a}\rangle|}{\|\hat{a}\|_1} \quad (8)$$

where $\tilde{x}_i=\psi(x_i),\tilde{x}_j=\psi(x_j)$, and x_i, x_j are the two samples in the original space.

Here, we explore the relation between the distance metric defined in (8) and the conditional probabilities in AdaBoost. Let $P(y|x)$ be a conditional probability of class labels given input x. Suppose that samples are identically and independently distributed. When the number of samples approaches infinity, the empirical risk converges in probability to the expectation of the cost function by the law of large numbers,

$$\frac{1}{N}\sum_{i=1}^{N}\exp(-y_i f(x_i))\to\int_x\sum_{y=\pm1}P(y|y)\exp(-yf(x))dx=E(\exp(-yf(x)))\quad(N\to\infty). \quad (9)$$

We derive predictor $f(x)$ that minimizes risk $E(\exp(-yf(x))$. The integrand of (9) is minimized at $f(x)$, which satisfies the equation

$$\frac{\partial\left[\sum_{y=\pm 1}P(y|x)\exp(-yf(x))\right]}{\partial f(x)}=0 \tag{10}$$

Then

$$\frac{P(1|x)}{P(-1|x)}=\frac{\exp(f(x))}{\exp(-f(x))}=\exp(2f(x))\Rightarrow f(x)=\frac{1}{2}\log\left(\frac{P(1|x)}{P(-1|x)}\right) \tag{11}$$

According to the distance metric defined in (8), we have

$$D(x_i,x_j)=\frac{|\langle\tilde{x}_i,\hat{a}\rangle-\langle\tilde{x}_j,\hat{a}\rangle|}{\|\hat{a}\|_1}=\frac{|f(x_i)-f(x_j)|}{\|\hat{a}\|_1} \tag{12}$$

and substituting (11) into (12), we can obtain

$$\begin{aligned}D(x_i,x_j)&=\frac{|f(x_i)-f(x_j)|}{\|\hat{a}\|_1}=\frac{1}{2\|\hat{a}\|_1}\left|\log\left(\frac{P(1|x_i)}{P(-1|x_i)}\right)-\log\left(\frac{P(1|x_j)}{P(-1|x_j)}\right)\right|\\&=\frac{\left|\log\left(\frac{P(1|x_i)}{P(-1|x_i)}\frac{P(-1|x_j)}{P(1|x_j)}\right)\right|}{2\|\hat{a}\|_1}\end{aligned} \tag{13}$$

Since $\|\hat{a}\|_1$ is a constant item, the distance metric defined in (8) denotes the similarity of the conditional probability between x_i,x_j, when the number of samples approaches infinity, and predictor $f(x)$ that minimizes risk $E(\exp(-yf(x))$.

In practice, a nearly infinite number of samples is impossible, so empirical risk does not converge in probability to the expectation of the cost function, and predictor $f(x)$ minimizes risk $(1/N)\sum_{i=1}^{N}\exp(-y_if(x_i)$ rather than $E(\exp(-yf(x))$. Thus $f(x)$ is only an approximate of $(1/2)\log(P(1|x)/P(-1|x))$. This leads to a problem, in that all the samples are neighborhoods of the estimation point, if these samples have the same l_1-margin as the estimation point. The distance metric defined in (8) thus has low resolution.

After J iterations, we represent $f(x)$ as

$$f(x)=[1,1,\ldots,1]\left[a_1h_1(x),a_2h_2(x),\ldots,a_Jh_J(x)\right]^T \tag{14}$$

Substituting (14) in (12), we can obtain

$$\begin{aligned}D(x_i,x_j)&=\frac{\left|[1,1,\ldots,1]\left(\left[a_1h_1(x_i),\ldots,a_Jh_J(x_i)\right]^T-\left[a_1h_1(x_j),\ldots,a_Jh_J(x_j)\right]^T\right)\right|}{\|\hat{a}\|_1}\\&=\frac{\left|\sum_{t=1}^{J}(a_th_t(x_i)-a_th_t(x_j))\right|}{\|\hat{a}\|_1}\end{aligned} \tag{15}$$

In this paper, we approximate $D(x_i,x_j)$ with $\tilde{D}(x_i,x_j)$,

$$\tilde{D}(x_i,x_j)=\frac{\sum_{t=1}^{J}|(a_th_t(x_i)-a_th_t(x_j))|}{\|\hat{a}\|_1}=\frac{\|f(x_i)-f(x_j)\|_1}{\|\hat{a}\|_1} \tag{16}$$

where $\|f(x_i)-f(x_j)\|_1$ denotes the l_1-norm. The distance metric thus defined, $\tilde{D}(x_i,x_j)$, has high resolution.

When $D(x_i,x_j)$ defined in (15) is used as a distance metric in weighted kNN algorithm, all the samples having the l_1-margin as the estimation sample are neighborhoods of the estimation point. Namely, weighted kNN produces highly stretched neighborhoods along boundary directions, thus the weighted kNN algorithm depends too much on AdaBoost which learned some prior knowledge from training samples. It is not surprising that poor performance of AdaBoost in some cases, such as underfitting or overfitting, will result in a significant degradation in classification accuracy for weighted kNN algorithm. However, the weighted kNN algorithm can reduce the excessive dependence on AdaBoost, if we approximate $D(x_i,x_j)$ with $\tilde{D}(x_i,x_j)$.

4.2. Identification of suspect samples

In this section, we present a new idea for the identification of suspect samples. In AdaBoost, after J iterations, some samples are misclassified. In noisy situations, we know that it is difficult for classifiers to capture noisy samples. So it is mainly suspect samples that are contained in these misclassified samples. Ideally, these misclassified samples can be divided into two different categories: good samples and suspect samples (mislabeled or atypical samples):

- Good samples are those samples on which AdaBoost should concentrate in the next iteration.
- A suspicious sample is a sample, for which the optimal prediction for a given loss and the family of classifiers, differs from its current label, or those observations regarding rare samples, which are often associated with variable asymmetry and are an important cause for leverage points. The influence of these samples should be eliminated in the next iteration [22].

After each iteration in AdaBoost, we call the weighted kNN to identify suspect samples. The feature space is composed of the hypothesis set after J iterations. A misclassified sample x_i is considered a good sample, if the k weighted neighborhood of x_i has a majority of the same label as x_i. In most of the edited k nearest neighbor rules, these samples are thought of as good samples. AdaBoost should concentrate on these misclassified samples in the next iteration. However, a misclassified sample is subject to caution, if the k weighted neighborhood of x_i has a majority of different labels to x_i. These samples are often associated with variable asymmetry and are an important cause for leverage points. Their influence should be eliminated in the next iteration.

In discussing the strategy for eliminating the influence of some samples, we analyze Fig. 1. In the illustration on the left, we have one atypical sample, which corrupts the estimation. AdaBoost will concentrate on this atypical sample to the detriment of the good estimate obtained by removing the atypical sample. In the illustration on the right, removing the mislabeled samples leads to AdaBoost generating a simple hypothesis, which is a good generalization.

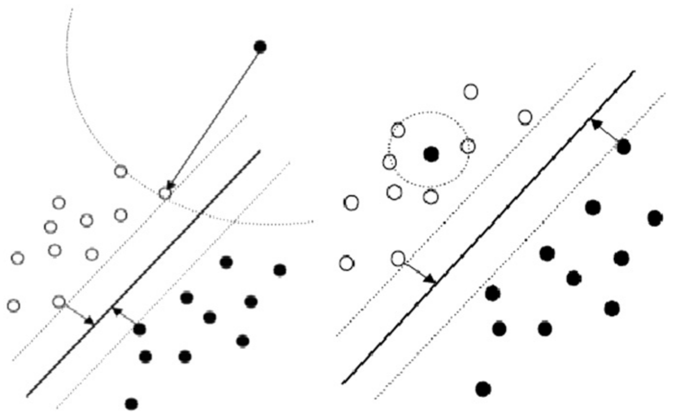


Fig. 1. Strategy for removing samples: data with one atypical sample (left) and with a mislabeled sample (right). The solid line shows the decision boundary when removing the suspect sample, while the dashed line marks the margin area.

The algorithm for connecting AdaBoost with the weighted kNN can associate the options of removal with the hypothesis generated by AdaBoost.

The new boosting algorithm avoids overfitting by eliminating the influence of some mislabeled samples (or atypical samples) in the next iteration. If $f(x)$ is too simple, there are higher approximation errors when $f(x)$ is used as an approximate of $(1/2)\log\left(\frac{P(1|x)}{P(-1|x)}\right)$. Most of the samples will be viewed as mislabeled samples (or atypical samples) by the weighted kNN. To avoid this problem, none of the samples is removed from the training set; we only modify the weight of the samples that are viewed as mislabeled samples (or atypical samples) by the weighted kNN. In other words, by setting the weight of these samples to zero, their harmful effects can be eliminated. After each AdaBoost iteration, samples that are considered to be mislabeled are redetermined by the weighted kNN. This design preserves and improves the adaptability of the new algorithm, especially in noisy domains. The EAdaBoost algorithm is described in Table 2.

To provide a better comprehension of EAdaboost and its capabilities, we perform numerical simulations on a toy dataset (banana dataset with 400 instances) with a number (150) of boosting steps. We firstly generate a partition into training and test set (mostly $\approx 2/3{:}1/3$). We then train a classifier and compute its test set error. The number of target neighbors k in the weighted kNN algorithm is set by cross validation. Decision tree with no more than 6 leaf nodes is used as the base learner. Graphical results are shown in Fig. 2.

The separating surface produced by AdaBoost (the upper-left plot) and EAdaBoost (the upper-right plot) is shown in Fig. 2. In the two subplots, the positive and negative training patterns are shown as '*' and 'o', respectively. And misclassified input patterns are marked with '■'. Regions classified as positive are colored blue and the negative are colored green. The lower-left plot illustrates the changes in training and test errors of AdaBoost as a function of the number of boosting steps, while the lower-right plot illustrates the changes in training and test errors of EAdaBoost.

As the number of learning cycles increases, the most difficult patterns are emphasized strongly, so, the AdaBoost algorithm achieves asymptotically a decision with hard margin as shown in our toy example (cf. Fig. 2). Fig. 2 illustrates that AdaBoost overfits after about 25 boosting steps and produces cluttered separating surface. While the EAdaBoost algorithm can associate the options of removal with the hypothesis generated by AdaBoost and produce smooth boundary. As the number of learning cycles increases, EAdaBoost consistently shows improvement for the training set.

Table 2
EAdaBoost algorithm (S, J, k).

```
Input: $S=(x_i,y_i), i=1,2,\ldots,N$, $x_i \in X$, $y_i \in \{-1,+1\}$
Initialize: $D_1(x_i)=(1/N)$, $(i=1,2,\ldots,N)$
Do for $t=1,2,\ldots,J$:
  1. Train classifier with respect to the weighted sample set $\{S, D_t\}$, and obtain the weak hypothesis $h_t$: $X \to \{-1,+1\}$
  2. Calculate the training error $\varepsilon_t$ of $h_t$: $\varepsilon_t = \sum_{i=1}^{N} D_t(x_i) I(h_t(x_i) \neq y_i)$
  3. Choose $a_t=(1/2)\log(1-\varepsilon_t/\varepsilon_t)$ then $f_t(x)=\sum_t a_t h_t(x)$
  4. Update the weights of the samples $D_{t+1}(x_i)=\frac{\exp(-y_i f_t(x_i))}{\sum_{i=1}^{N}\exp(-y_i f_t(x_i))}$
  5. Let $\psi(x)=[h_1(x),h_2(x),\ldots,h_t(x)]$, and let the diagonal matrix $\Lambda$ with
     $\begin{cases}\Lambda_{i,j}=a_i & \text{if } i=j\\ \Lambda_{i,j}=0 & \text{if } i\neq j\end{cases}$ where $t \geq i \geq 0$, $t \geq j \geq 0$. Call weighted kNN Algorithm ($S$,
     $\psi$, $\Lambda$, $k$), and get $g(x)=sgn(ave_{\tilde{x}_i \in R(x')} y_i)$, where $\tilde{x}_i=\Lambda\psi^T x_i$ and $x_i \in S$
  A. Let $S'_t=\{(x_i,y_i)\in S, \rho_t(x_i,y_i)<0\}$
  B. if $(x_i, y_i)\in S'_t$ and $y_i \neq g(x_i)$, let $D_{t+1}(x_i)=0$; normalized $D_{t+1}(x_i)$
Output the final hypothesis: $f_T(x)=\sum_{t=1}^{J}(a_t/\sum_{r=1}^{J} a_r)h_t(x)$
```

4.3. Some analyses

In Section 4.2, we presented an algorithm that connects AdaBoost with the weighted kNN to remove some samples. To understand how the algorithm improves prediction accuracy, we introduce some analyses.

In pattern classification problems, learning algorithms aim to choose a classifier with a small error and high generalization capability. AdaBoost does this by optimizing an exponential cost function that is defined in terms of the training data. This cost function can be thought of as an error estimate:

$$\min_{f(x)} \frac{1}{N}\sum_{i}^{N} \exp(-y_i f(x_i)) \tag{17}$$

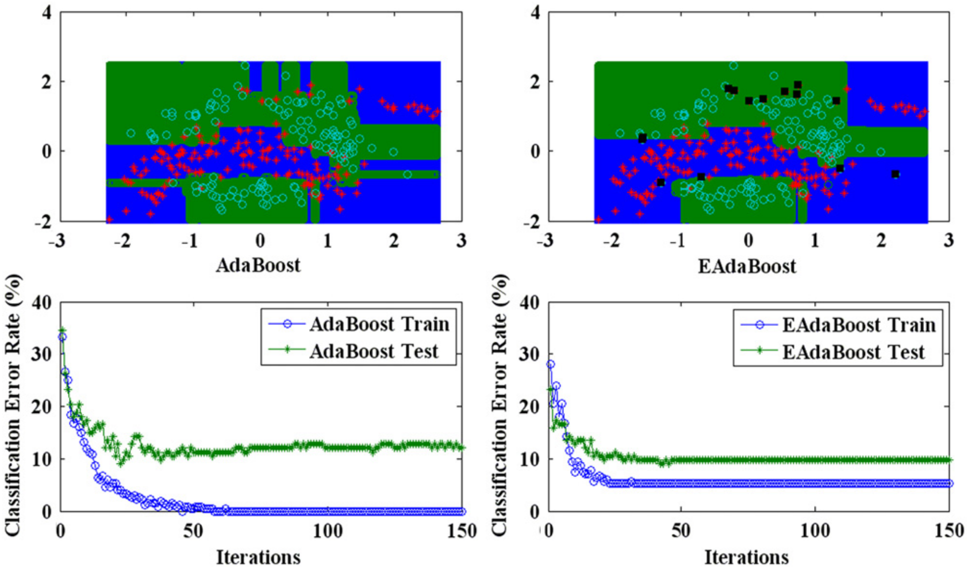


Fig. 2. Graphical results for a two dimensional classification problem. From left to right: separating surfaces and classification error rate of AdaBoost; separating surfaces and classification error rate of EAdaBoost.

where $f(x)$ is the current classifiers ensemble. In the case of probabilistic setup, the expectation of loss [15] is

$$E(\exp(-yf(x))) = \int_x \sum_{y=\pm 1} P(y|x)\exp(-yf(x))\,dx \tag{18}$$

In EAdaBoost, when there are finite samples in S, the posterior probability $P(y|x)$ of the observation point x is estimated by Eq. (2). Then,

$$\begin{aligned}\frac{1}{N}\sum_i^N \exp(-y_i f(x_i)) &= \sum_{x\in S}\left(\frac{n_x}{N}\times\frac{1}{n_x}\sum_{x_i=x}(\exp(-y_i f(x_i))\right)\\ &=\sum_{x\in S}\frac{n_x}{N}\times\left(\frac{1}{n_x}\sum_{x_i=x,y_i=+1}(\exp(-f(x_i))+\frac{1}{n_x}\sum_{x_i=x,y_i=-1}(\exp(f(x_i))\right)\\ &=\sum_{x\in S}\frac{n_x}{N}\times\left(\frac{n_{x,+1}}{n_x}\exp(-f(x_i))+\frac{n_{x,-1}}{n_x}\exp(f(x_i))\right)\\ &=\sum_{x\in S}\frac{n_x}{N}\times\left(\frac{k[1]}{k}\exp(-f(x_i))+\frac{k[-1]}{k}\exp(f(x_i))\right)\end{aligned} \tag{19}$$

where $n_{x,+1}, n_{x,-1}$ denote the number of samples $(x,+1)$ and $(x,-1)$, respectively.

Consider the 'risk' of a fixed instance x in S (because after each AdaBoost iteration suspect samples are mostly misclassified, we focus primarily on misclassified samples):

$$r(x) = \frac{k[1]}{k}\exp(-f(x))+\frac{k[-1]}{k}\exp(f(x)) \tag{20}$$

where usually $n_x=1$. For AdaBoost,

$$r(x) = \exp(-yf(x)) \tag{21}$$

In this case minimizing the risk of a suspect sample actually increases the true expectation of loss for the fixed instance. For the Boosting algorithm, minimizing average loss means forcing the classifier to fit all training samples, including the suspect samples [19]. Compared with AdaBoost, for EAdaBoost (for simplicity, we consider a relabeling strategy for misclassified samples):

$$r(x) = \exp\left(-sgn\left(ave_{x_i\in R(x)}y_i\right)f(x)\right) \tag{22}$$

where $R(x)$ is a local subregion, and $R(x)$ is obtained by the weighted kNN.

By Eq. (2), the posterior probability $P(1|x)$ of the observation point x is estimated by kNN to be $(k[1]/k)$. Then the boundary bias of a fixed instance x in S is

$$b\left(E_S\left(\frac{k[1]}{k}\right),\tilde{f}(x)\right) = sgn(1/2-\tilde{f}(x))\left(E_S\left(\frac{k[1]}{k}\right)-1/2\right) \tag{23}$$

where E_S denotes the expectation taken with respect to the repetition sampling training set S, and $\tilde{f}(x)$ is the result associated with the Bayesian classifier $(\tilde{f}(x)\doteq P(y=1|x))$. Domingos [35] showed that so long as boundary bias is negative $b\left(E_S(k[1]/k),\tilde{f}(x)\right)<0$ classification error decreases with increasing $|E_S(k[1]/k)-1/2|$ irrespective of the estimation bias $\left|E_S(k[1]/k)-\tilde{f}(x)\right|$. Friedman [36] showed that the kNN procedures are produced by over-smoothing, boundary bias tends to be negative, namely $b\left(E_S(k[1]/k),\tilde{f}(x)\right)<0$, and decreasing the variance (increasing margins) can have a dramatic impact on reducing boundary error. In Eq. (22), we substitute label y of the misclassified samples with $sign(ave_{xi\in R(x)}y_i)$, which can make sure the boundary bias of a fixed instance x in S tends to be negative. The main advantage of AdaBoost over other boosting techniques is that it is able to take advantage of weak hypotheses to maximize the minimum margin, thus, the EAdaBoost where AdaBoost and kNN naturally complement each other can resist overfitting and produce a good generalization.

5. Experiments

In the previous sections, we discussed how EAdaBoost tackles the problem of overfitting theoretically. In this section, we examine empirically whether our new algorithm performs better than the others.

For the experiments presented in this paper, we used a selection of two-class data sets from the UC Irvine database [37]. These data sets are listed in Table 3.

At first each data set was randomly split into a training set (2/3 of the original dataset), and a test set (1/3) for each run, we trained a classifier and then computed its test set error on each run. All of the experimental results were averaged over 30 runs of the randomly generated splits of the data. Both the number of target neighbors k in the weighted kNN algorithm and the parameter C of weight decay algorithm mentioned in Sections 5.2 and 5.4 were set by cross validation. (For the purpose of cross validation, the training sets were further randomly partitioned into training (2/3 of the training sets) and validation sets (1/3).)

For fair comparison, we used a fixed number of iterations for all algorithms [7]. In all experiments, we set the maximum training iteration to 300, and a decision tree with no more than 6 leaf nodes was used as the base classifier.

We also studied the robustness of the new algorithm by introducing labeling noise to the ten data sets. We randomly selected some of the samples and reversed their labels while leaving the other samples unchanged. In this way, Noise-UCI data sets with 10%, 20%, and 30% random noise were constructed. The same method was used to construct noisy data in [38].

5.1. Measuring the quality of weighted kNN algorithm

First we examined the general effectiveness of the weighted kNN algorithm using the distance metric designed in this paper by comparing it with the kNN in the original space using Euclidean distance. AdaBoost was run firstly on the training data to produce a sequence of base classifiers and their corresponding weights. The weighted kNN algorithm was then run on the representation space composed of the classifiers produced by AdaBoost. Table 4 gives the results of our experiments, with the smaller error, in each case, shown in bold.

From Table 4, it is clear that the weighted kNN algorithm is able to achieve lower classification errors than kNN for all the ten data sets, whether or not noise is present. In [25,33] it is shown that kNN classification can be significantly improved when cleverly combined with prior knowledge. Even a simple (global) linear transformation of input features has been shown to yield

Table 3
Description of data sets.

Collection name	No. of instances	No. of attributes
Breast-Cancer	699	10
German	419	25
Heart-data	270	14
Hepatitis	155	20
Sonar	208	61
Wdbc	569	31
Wpbc	198	34
Ionosphere	351	34
Tic-Tac-Toe	958	10
Pima Indians Diabetes	768	9

Table 4
Results of the experiments with noise.

Data	0% noise		10% noise		20% noise		30% noise	
	kNN	Weighted kNN	kNN	Weighted kNN	kNN	Weighted kNN	kNN	Weighted kNN
Breast-Cancer	0.04449	**0.04092**	0.14537	**0.13935**	0.23893	**0.21462**	0.32534	**0.31761**
German	0.27857	**0.25714**	0.35	**0.27143**	0.37143	**0.3**	0.47857	**0.31429**
Heart-data	0.31111	**0.2**	0.392	**0.23333**	0.41511	**0.27778**	0.42815	**0.36311**
Hepatitis	0.451	**0.43547**	0.45298	**0.44118**	0.46712	**0.44834**	0.4852	**0.45604**
Sonar	0.22019	**0.16058**	0.26087	**0.2029**	0.30435	**0.27536**	0.37692	**0.34375**
Wdbc	0.05790	**0.02632**	0.1	**0.04211**	0.10526	**0.06842**	0.18421	**0.16316**
Wpbc	0.30303	**0.25758**	0.33081	**0.30051**	0.34848	**0.33636**	0.3899	**0.37818**
Ionosphere	0.11111	**0.05983**	0.12821	**0.07692**	0.13675	**0.12821**	0.23932	**0.16239**
Tic-Tac-Toe	0.22571	**0.01254**	0.22571	**0.1442**	0.24138	**0.2163**	0.29467	**0.24765**
Pima Indians Diabetes	0.28516	**0.22656**	0.29297	**0.23047**	0.32031	**0.28125**	0.38281	**0.30859**

much better kNN classifiers. In this paper, the representation space composed of classifiers produced by AdaBoost, can capitalize on some statistical regularity in the data and all similarly labeled inputs are clustered in the same subspace in the feature space. AdaBoost is able to take advantage of weak hypotheses to maximize the minimum margin. The weighted kNN algorithm (including the constitution of the representation space and the distance metric designed in this paper) ensures that the k-nearest neighbors always belong to the same class with a similar conditional probability, while samples from different classes are separated by a large margin.

5.2. *Standard evaluation of EAdaBoost*

Each experiment consisted of the following steps. First, AdaBoost was run on the training data to produce base classifiers and their corresponding weights. After each AdaBoost iteration, the weighted kNN algorithm was run on the representation space, to identify suspect samples. The influence of samples considered as suspect samples by the weighted kNN are eliminated in the next iteration of AdaBoost.

Many studies have been conducted on the issue of overfitting in the AdaBoost algorithm. However, most of the modified algorithms are unsuccessful due to either the high computational cost or lack of strong empirical results showing improvement [38]. One of the most successful algorithms is the weight decay method which is widely used in neural networks [39,40] and support vector machines. The basic idea of the weight decay method is to add slack variables into the cost function to reduce the influence of ‘hardness’ samples. The weight decay method has been shown to achieve some improvement over the AdaBoost algorithm. We examined the general effectiveness of the new Boosting algorithm by comparing it with AdaBoost and weight decay. The experimental procedure was repeated 30 times, and the mean and standard deviation of the test errors are presented in Table 5. The lowest error in each case is shown in bold.

From Table 5, we can see that EAdaBoost is able to achieve lower classification errors than AdaBoost for all of the ten data sets. And EAdaBoost achieves the best performance in 8/10 of the data sets; and in the remaining two its error rate is the same as weight decay. To further test the robustness of the new algorithm, the standard deviation of the test error over 30 runs, is also presented in the table. The standard deviation of EAdaBoost is roughly equal to that of AdaBoost, but less than that of weight decay in eight out of the ten data sets. This indicates that EAdaBoost is more robust or adaptive than weight decay in most of the cases. Based on the above observations, the techniques of EAdaboost help to improve the effectiveness of the Boosting algorithm.

Table 5
Effectiveness of boosting algorithms.

Data	AdaBoost		EAdaBoost		Weight decay	
	Mean	Std.	Mean	Std.	Mean	Std.
Breast-Cancer	0.04735	0.0083	**0.03562**	0.0078	0.04220	0.0150
German	0.29286	0.0354	**0.25714**	0.0326	**0.25714**	0.1640
Heart-data	0.18889	0.0347	**0.13333**	0.0366	0.15556	0.0229
Hepatitis	0.45098	0.0549	**0.31373**	0.0595	0.43137	0.0873
Sonar	0.13043	0.0443	**0.10145**	0.0344	0.17391	0.0600
Wdbc	0.02632	0.0110	**0.02105**	0.0208	0.04211	0.0127
Wpbc	0.25758	0.0587	**0.21212**	0.0395	0.25758	0.0737
Ionosphere	0.06838	0.0166	**0.05983**	0.0167	0.07692	0.0222
Tic-Tac-Toe	0.00031	0.0054	**0.00010**	0.0048	**0.00010**	0.0186
Pima Indians Diabetes	0.26172	0.0244	**0.21484**	0.0234	0.25391	0.0238

Fig. 3 illustrates the changes in training and test errors as a function of the number of learning cycles for the Breast-Cancer and Hepatitis data sets. From Fig. 3, we can see that the AdaBoost and weight decay algorithm suffers from overfitting for the Breast-Cancer and Hepatitis as the number of learning cycles increases, while EAdaBoost consistently shows improvement for the two data sets. EAdaBoost in the convergence rate in training samples significantly faster than weight decay algorithm, and after finite iteration, EAdaBoost train error does not decline. EAdaBoost avoid concentrating its resources on a few suspect samples.

5.3. *Margins*

Mason et al. [40] showed that the value of the minimum training margin has no real impact on generalization performance. They also stated that it is more important for generalization performance to find a balance between training error and complexity. The effect of EAdaBoost is best illustrated by comparing the cumulative margin distributions generated by AdaBoost, EAdaBoost and the weight decay method. Fig. 4 shows the comparisons for Breast-Cancer and Hepatitis data sets. For a given margin the value on the curve corresponds to the proportion of samples with margin less than or equal to this value.

Fig. 4 shows that in trying to increase the margins of negative samples, AdaBoost is willing to sacrifice a significant portion of the margin of positive samples. In contrast, the new Boosting algorithm ‘gives up’ on samples with a large negative margin. The same result can be observed in weight decay algorithm. Combined with the changes in training and test errors (illustrated in Fig. 3), we can see that the generalization performance of the combined

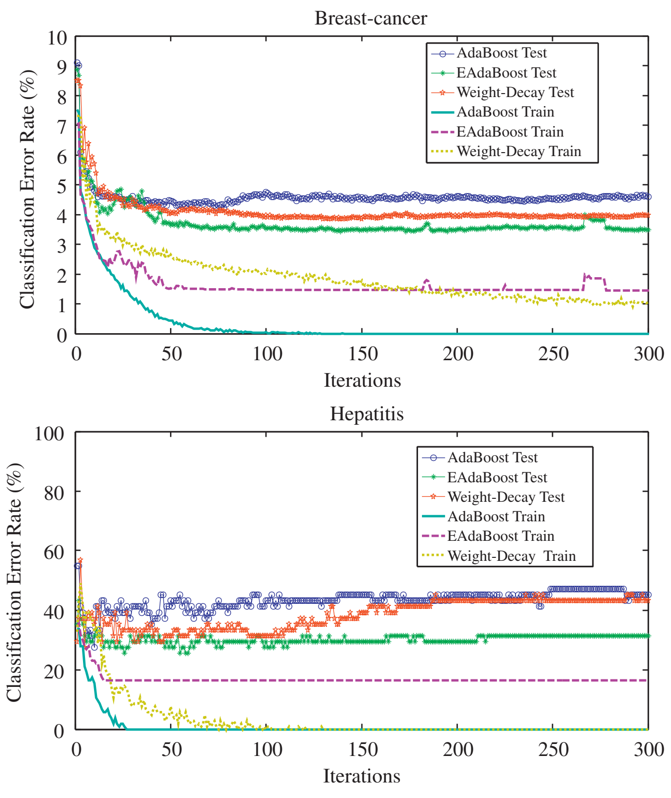


Fig. 3. Test and training error curves for EAdaBoost, AdaBoost, and weight decay.

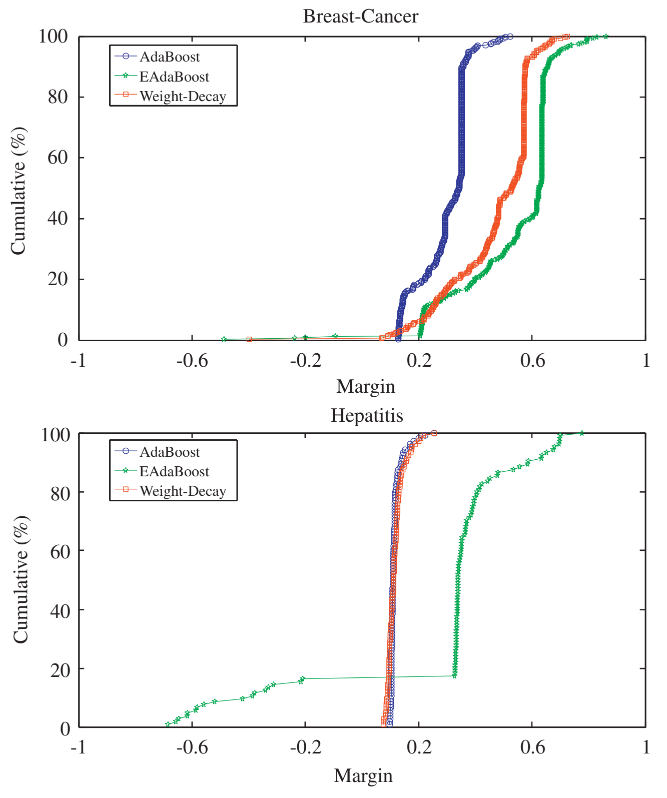


Fig. 4. Margin distributions for EAdaBoost, AdaBoost, and weight decay.

classifier produced by EAdaBoost can be as good as or better than that of the classifier produced by AdaBoost and weight decay algorithm, despite having a dramatically worse minimum training margin.

Table 6
Mean and standard deviation of test error on UCI data sets with 10% label noise.

Data	10% noise					
	AdaBoost		EAdaBoost		Weight decay	
	Mean	Std.	Mean	Std.	Mean	Std.
Breast-Cancer	0.08761	0.01191	**0.04831**	0.0083	0.08188	0.0183
German	0.27857	0.0358	**0.22857**	0.0375	0.26429	0.1548
Heart-data	0.24444	0.0481	**0.2**	0.0573	0.23333	0.0477
Hepatitis	0.35294	0.0534	**0.27451**	0.0594	0.31373	0.0472
Sonar	0.18841	0.0547	**0.14493**	0.0482	0.2029	0.0460
Wdbc	0.07895	0.0260	**0.06316**	0.0291	0.07368	0.0188
Wpbc	0.30303	0.0538	**0.28788**	0.0382	0.31818	0.1153
Ionosphere	0.12282	0.0308	**0.05983**	0.0216	0.11111	0.0237
Tic-Tac-Toe	0.10658	0.0198	**0.10345**	0.0395	0.16301	0.1359
Pima Indians Diabetes	0.32031	0.0330	**0.26953**	0.0258	0.27344	0.0377

Table 7
Mean and standard deviation of test error on UCI data sets with 20% label noise.

Data	20% noise					
	AdaBoost		EAdaBoost		Weight decay	
	Mean	Std.	Mean	Std.	Mean	Std.
Breast-Cancer	0.13662	0.0150	**0.08269**	0.0209	0.17893	0.0276
German	0.32857	0.0516	**0.22857**	0.0354	0.31429	0.1869
Heart-data	0.27778	0.0501	**0.25778**	0.0575	0.26667	0.0535
Hepatitis	0.32741	0.0644	**0.26778**	0.0614	0.2763	0.0511
Sonar	0.30435	0.0651	**0.27536**	0.0565	0.34783	0.0570
Wdbc	0.14737	0.0324	**0.06842**	0.0246	0.09474	0.0236
Wpbc	0.37879	0.0581	**0.24242**	0.0578	0.34848	0.1524
Ionosphere	0.18803	0.0395	**0.10256**	0.0473	0.12821	0.0488
Tic-Tac-Toe	0.23152	0.0271	**0.1348**	0.0451	0.20313	0.1143
Pima Indians Diabetes	0.38281	0.0316	**0.26953**	0.0289	0.34375	0.0731

5.4. Robustness to noise

We further tested the robustness of EAdaBoost by comparing it with AdaBoost and the weight decay method in noisy situations. Noise was introduced to the ten data sets drawn from the UCI Machine Learning Repository. The results of the experiment are given in Tables 6–8.

As the number of learning iterations increases, the AdaBoost algorithm suffers from overfitting for most data sets. Moreover, weight decay is also prone to overfitting despite the fact that slack variables are introduced into the cost function to reduce the influence of hard-to-learn samples. On the other hand, EAdaBoost shows improvement for all of the ten data sets. In addition, the standard deviation demonstrates that our new algorithm exhibits great robustness to noise.

Another result can be noted from Tables 7 and 8, that is, the classification errors for weight decay are much greater than those for AdaBoost and EAdaBoost on the Breast-Cancer data sets with 20% and 30% noise. This phenomenon is actually anticipated, because parameter C for weight decay needs to be given in advance. When C is too small, the decline in training error for weight decay is too slow. Furthermore, when C is too large, sharp changes in the classification errors for weight decay take place during the iterative process. The effect of C is then best illustrated by Figs. 5 and 6.

Figs. 5 and 6 illustrate the changes in training and test errors as a function of the number of learning cycles for the Breast-Cancer data set with 20% and 30% noise, respectively. In the weight decay algorithm, sharp changes take place in the classification errors, but EAdaBoost shows stable improvement for the data sets.

Table 8
Mean and standard deviation of test error on UCI data sets with 30% label noise.

Data	30% noise					
	AdaBoost		EAdaBoost		Weight decay	
	Mean	Std.	Mean	Std.	Mean	Std.
Breast-Cancer	0.20067	0.0223	**0.15603**	0.0246	0.29238	0.0290
German	0.38571	0.0412	**0.30714**	0.0505	0.40714	0.1314
Heart-data	0.47778	0.0601	**0.43333**	0.0700	0.52222	0.0747
Hepatitis	0.46863	0.0569	**0.43137**	0.0932	**0.43137**	0.0677
Sonar	0.31884	0.0625	**0.21739**	0.0690	0.23188	0.0820
Wdbc	0.27895	0.0399	**0.06316**	0.0454	0.25263	0.0502
Wpbc	0.33333	0.0519	**0.24242**	0.0727	**0.24242**	0.1787
Ionosphere	0.28432	0.0421	**0.25641**	0.0460	0.26496	0.0469
Tic-Tac-Toe	0.24138	0.0300	**0.2069**	0.0649	0.42633	0.1198
Pima Indians Diabetes	0.4375	0.0424	**0.31641**	0.0368	0.375	0.0443

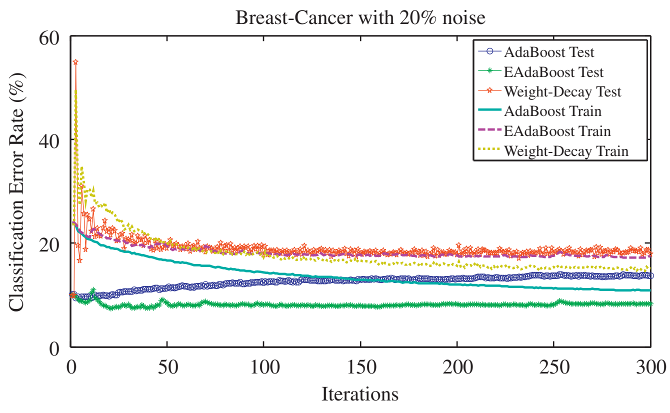


Fig. 5. Test and training error curves for EAdaBoost, AdaBoost, and weight decay with 20% noise.

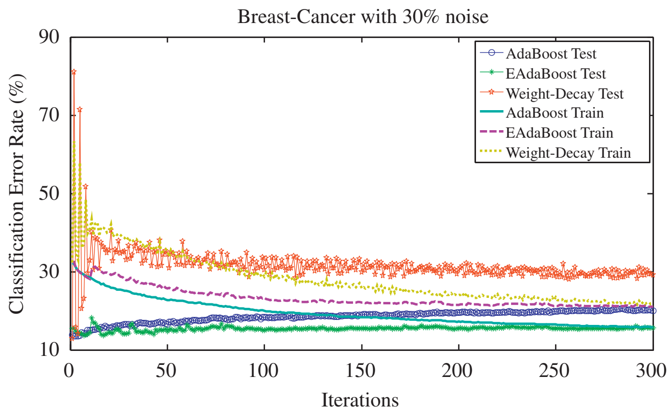


Fig. 6. Test and training error curves for EAdaBoost, AdaBoost, and weight decay with 30% noise.

6. Conclusions

Making boosting methods more robust against label noise and avoiding overfitting are interesting and well known topics that have been debated since the first algorithms were developed. Many important results have been obtained by researchers. To avoid overfitting, two problems need to be solved: (1) which samples can be dealt with as suspect samples, and (2) how many samples can be dealt with as suspect samples. Many boosting-type algorithms solve these problems implicitly. These algorithms analyze the dynamic evolution of AdaBoost weights for estimating 'hardness' of every training sample. Then overfitting is avoided by reducing the influence of some hard-to-learn samples. The other algorithms solve the two problems explicitly, although this requires knowing important additional parameters in advance. In this paper, a new algorithm, called EAdaBoost, has been designed, which solves the two problems explicitly and does not need any important additional parameters a priori. First, a weighted kNN algorithm was designed to distinguish between suspect and non-suspect samples explicitly. Then AdaBoost is used to enhance the classification accuracy and avoid overfitting by editing data sets using the weighted kNN algorithm for improving the quality of training data. In each iteration of AdaBoost, these suspect samples are dealt with skillfully to preserve and improve the adaptability of the new algorithm. Various numerical tests were performed on UCI data sets. The testing results show that the method presented in this paper is both efficient and effective.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable suggestions that improved this work. This work was partially supported by the National Natural Science Foundation of China (No. 60974101), Ph.D. Programs Foundation of Ministry of Education of China (No. 20060698029) and Program for New Century Excellent Talents in University (No. NCET-06-0828).

References

[1] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1) (1997) 119–139.
[2] R.E. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Machine Learning 37 (3) (1999) 297–336.
[3] H. Drucker, C. Cortes, L.D. Jackl, Y. LeCun, V. Vapnik, Boosting and other ensemble methods, Neural Computation 6 (6) (1994) 1289–1301.
[4] R.E. Schapire, Y. Freund, P.L. Bartlett, W.S. Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, Annals of Statistics 26 (5) (1998) 1651–1686.
[5] V. Gomez-Verdejo, M. Ortega-Moral, J. Arenas-Garcia, A.R. Figueiras-Vidal, Boosting by weighting critical and erroneous samples, Neurocomputing 69 (2006) 679–685.
[6] W. Jiang, Does boosting overfit: views from an exact solution, Technical Report 00-03, Department of Statistics, Northwestern University, 2000.
[7] G. Rätsch, T. Onoda, K.R. Müller, Soft margins for AdaBoost, Machine Learning 42 (3) (2001) 287–320.
[8] T.G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, Machine Learning 40 (2) (1999) 139–157.
[9] A.J. Grove, D. Schuurmans, Boosting in the limit: maximizing the margin of learned ensembles, in: Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98), Madison, WI, 1998, pp. 692–699.
[10] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting, Annals of Statistics 28 (2) (2000) 337–407.
[11] H.H. Bauschke, J.M. Borwein, Legendre functions and the method of random Bregman projections, Journal of Convex Analysis 4 (1) (1997) 27–67.
[12] E. Romero, L. Marquez, X. Carreras, Margin maximization with feed-forward neural networks: a comparative study with SVM and AdaBoost, Neurocomputing 57 (2004) 313–344.
[13] S. Abney, R.E. Schapire, Y. Singer, Boosting applied to tagging and pp attachment, in: Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, University of Maryland, USA, 1999, pp. 38–45.
[14] G. Martinez-Munoz, A. Sanchez-Martinez, D. Hernandez-Lobato, A. Suarez, Class-switching neural network ensembles, Neurocomputing 71 (2008) 2521–2528.
[15] R.A. Servedio, Smooth boosting and learning with malicious noise, Journal of Machine Learning Research 4 (2003) 633–648.
[16] G. Rätsch, T. Onoda, K. Muller, Regularizing AdaBoost, in: Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems, vol. II, Cambridge, MA, 1999, pp. 564–570.
[17] T. Kanamori, P. Takenouchi, P. Eguchi, N. Murata, Robust loss functions for boosting, Neural Computation 19 (8) (2007) 2183–2244.
[18] C. Domingo, O. Watanabe, A modification of AdaBoost, in: Proceedings of the COLT, Morgan Kaufmann, San Francisco, 2000, pp. 180–189.

[19] A. Vezhnevets, O. Barinova, Avoiding boosting overfitting by removing confusing samples, in: Proceedings of the 18th European Conference on Machine Learning (ECML), Warsaw, Poland, 2007, pp. 430–441.
[20] A. Angelova, Y. Abu-Mostafam, P. Perona, Pruning training sets for learning of object categories, in: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, USA, 2005, pp. 494–501.
[21] S. Merler, B. Caprile, C. Furlanello, Bias-variance control via hard points shaving, International Journal of Pattern Recognition and Artificial Intelligence 18 (5) (2004) 891–904.
[22] F. Muhlenbach, S. Lallich, D.A. Zighed, Identifying and handling mislabelled instances, Intelligent Information Systems 22 (1) (2004) 89–109.
[23] Y. Freund, An adaptive version of the boost by majority algorithm, Machine Learning 43 (3) (2001) 293–318.
[24] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1) (1967) 21–27.
[25] S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, USA, 2005, pp. 539–546.
[26] C. Domeniconi, D. Gunopulos, J. Peng, Large margin nearest neighbor classifiers, IEEE Transactions on Neural Networks 16 (4) (2005) 899–909.
[27] M.P. Kumar, P.H.S. Torr, A. Zisserman, An invariant large margin nearest neighbour classifier, in: Proceedings of the 11th IEEE International Conference on Computer Vision (ICCV'07), Rio de Janeiro, Brazil, 2007, pp. 1–8.
[28] D. Masip, J. Vitrià, Boosted discriminant projections for nearest neighbor classification, Pattern Recognition 39 (2) (2006) 164–170.
[29] F. Muhlenbach, S. Lallich, D.A. Zighed, Identifying and handling mislabelled instances, Journal of Intelligent Information Systems 22 (1) (2004) 89–109.
[30] R. Duda, P. Hart, in: Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[31] C.M. Bishop, in: Neural Networks for Pattern Recognition, Oxford University Press Inc., New York, 1995.
[32] R. Gil-Pita, X. Yao, Using a genetic algorithm for editing *k*-nearest neighbor classifiers, in: Intelligent Data Engineering and Automated Learning (IDEAL'07), Proceedings of the Eighth International Conference, Birmingham, UK, 2007, pp. 1141–1150.
[33] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, Journal of Machine Learning Research 10 (2009) 207–244.
[34] P.J. García-Laencina, J.L. Sancho-Gómez, A.R. Figueiras-Vidal, M. Verleysen, K nearest neighbours with mutual information for simultaneous classification and missing data imputation, Neurocomputing 72 (2009) 1483–1493.
[35] P. Domingos, A unified bias-variance decomposition for zero-one and squared loss, in: Proceedings of the 17th National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, Texas, USA, 2000, pp. 564–569.
[36] J.H. Friedman, On bias, variance, 0/1-loss, and the curse-of-dimensionality, Data Mining and Knowledge Discovery 1 (1) (1997) 55–77.
[37] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, 1998.
[38] R. Jin, Y. Liu, S.I. Luo, J. Carbonell, A.G. Hauptmann, A. New Boosting, Algorithm using input-dependent regularizer, in: Proceedings of the 20th International Conference on Machine Learning (ICML'03), Washington, DC, 2003, pp. 148–156.
[39] R. Meir, G. Rätsch, in: An introduction to boosting and leveraging, in: Advanced Lectures on Machine Learning (LNAI2600), Springer, Berlin, 2003, pp. 118–183.
[40] L. Mason, P.L. Bartlett, J. Baxter, Improved generalization through explicit optimization of margins, Machine Learning 38 (3) (2000) 243–255.

Yunlong Gao received the M.S. degree in School of Information Science and Engineering, Lanzhou University, Gansu, China, in March 2005. He is currently doing the Ph.D. degree in System Engineering in Systems Engineering Institute, Xi'an Jiaotong University. His research interests include machine learning, pattern recognition, ensemble learning algorithms, power system optimization, and scheduling.

Feng Gao received his B.S. in Information and Control Engineering, Xi'an Jiaotong University, China in 1988, and his M.S. and Ph.D. in Systems Engineering from the same university in 1991 and 1996. He is currently a professor in Systems Engineering Institute, Xi'an Jiaotong University. His research interests include intelligent control, neural networks, power system optimization, and scheduling.