SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

December 2021, Vol. 64 222101:1–222101:11
https://doi.org/10.1007/s11432-019-2739-2

A clustering-based flexible weighting method in AdaBoost and its application to transaction fraud detection

Chaofan YANG[1,2], Guanjun LIU[1,2*], Chungang YAN[1,2] & Changjun JIANG[1,2*]

[1]*Department of Computer Science, Tongji University, Shanghai* 201804*, China;*
[2]*Shanghai Electronic Transactions and Information Service Collaborative Innovation Center, Tongji University, Shanghai* 201804*, China*

Received 26 April 2019/Revised 4 August 2019/Accepted 26 December 2019/Published online 25 November 2021

Abstract AdaBoost is a famous ensemble learning method and has achieved successful applications in many fields. The existing studies illustrate that AdaBoost easily suffers from noisy points, resulting in a decline of classification performance. The main reason is that it increases the weights of all misclassified samples (especially noisy points) in the same way so that the influence of noisy points can hardly be weakened. In this paper, the clustering algorithm is used to dynamically decide noisy points in the process of iterations. More precisely, we compute a misclassification degree for every cluster in every iteration that is used to decide if a misclassified sample is a noisy point or not in the current iteration. Furthermore, we propose a flexible method to update the weights of the misclassified samples. The experimental results on 22 public datasets show that our method achieves better results than the state-of-the-art methods including AdaBoost, AdaCoast, LogitBoost, and SPLBoost. We also apply our method to the transactions fraud detection, and the experiments on our real big dataset of transactions also illustrate its good performance.

Keywords ensemble learning, AdaBoost, clustering, misclassification degree, transaction fraud detection

Citation Yang C F, Liu G J, Yan C G, et al. A clustering-based flexible weighting method in AdaBoost and its application to transaction fraud detection. Sci China Inf Sci, 2021, 64(12): 222101, https://doi.org/10.1007/s11432-019-2739-2

1 Introduction

Machine learning technology has been widely used in practice. For example, the method of pattern classification has achieved good results in the fields of spam classification and processing [1], network intrusion detection [2], face recognition [3], and financial fraud detection [4]. As the application scenarios increase and data are continuously enriched, a single machine learning method (such as decision tree [5], support vector machine [6] or naive Bayes [7]) can hardly achieve an ideal classification effect. Thus, the combination of classifiers which is named ensemble learning [8, 9] becomes a new research direction. Ensemble learning obtains a higher classification accuracy by combining multiple weak classifiers.

At present, the ideas of ensemble learning mainly include Boosting [10] and Bagging [11]. We focus on the Adaboost algorithm in boosting learning [12]. AdaBoost is a classic classification algorithm whose main advantage lies in its adaptability. The kernel of AdaBoost is that if a sample is misclassified in an iteration, then its weight will be increased in the next iteration; if a sample is correctly classified, then the weight of the sample will be reduced in the next iteration. By repeating the training on the same dataset with different weight distributions, a series of weak classifiers are produced and then linearly combined to form a strong classifier with a low classification error rate.

A traditional AdaBoost algorithm [12] does not consider distribution of the dataset in the feature space and deals with all misclassified samples in the same way. This easily makes the outlier misclassified samples' weights be over-diffused after multiple iterations, i.e., the classifier pays more attention to those outlier misclassified samples and thus affects the classification performance. Some methods have been

* Corresponding author (email: liuguanjun@tongji.edu.cn, cjjiang@tongji.edu.cn)

proposed to deal with the problem. Wei et al. [13] used cost-sensitive learning to improve the AdaBoost algorithm named AdaCost. Friedman et al. [14] proposed LogitBoost which applied the Newton step to fit an additive symmetric logistic replica by maximizing the likelihood and minimizing the logistic loss. Wang et al. [15] proposed SPLBoost algorithm based on Self-paced learning, which integrated the robust Self-paced learning idea into the learning framework. These algorithms have achieved better experimental results than AdaBoost and their idea is mainly to reduce the influence of noisy points. However, they do not explicitly identify the noisy points in the training process. In this paper, we propose a method to dynamically identify noisy points in the training process and then present a strategy to flexibly update the weights of misclassified samples in view of the noisy points.

Our algorithm is called as CAdaBoost (clustering-based AdaBoost) because we first use a clustering method to divide the training set. In every iteration, we calculate a misclassification degree for each cluster. Then, we utilize the misclassification degree of a cluster to decide the possibility that a misclassified sample in the cluster is a noisy point. In each iteration, a flexible strategy of updating the weight of a misclassified sample is taken based on the decision of noisy points. Lots of experiments illustrate that our CAdaBoost which dynamically decides noisy points and flexibly updates the weights of misclassified samples can enhance the classification performance compared to the state-of-the-art improved AdaBoost. We also apply our method to solve the problem of checking transaction fraud in a financial company and obtain a good result.

The rest of this article is organized as follows. Section 2 introduces CAdaBoost in details. Section 3 introduces the to-be-compared methods and experimental results on 22 public datasets. Section 4 applies our CAdaBoost to the problem of transaction fraud detection. Section 5 summarizes the paper briefly.

2 CAdaBoost

This section first reviews Adaboost and clustering methods, and then describes our method CAdaBoost.

2.1 AdaBoost algorithm

The purpose of AdaBoost is to learn a series of weak classifiers from the training data and then combines them into a strong classifier. $X = \mathbb{R}^d$ represents the d-dimensional sample space and $Y = \{-1, +1\}$ represents the label set, where $y = +1$ represents a positive sample and $y = -1$ represents a negative sample. $D = \{(x_1,\ y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ denotes a training set where $x_i \in X$ represents one sample and $y_i \in Y$ represents the label corresponding to x_i. The AdaBoost algorithm mainly includes three steps [12].

Step 1. Initialize the weight distribution of the training data such that all training samples have the same weight at the beginning,

$$D_1 = \ (\omega_{1,1}, \omega_{1,2}, \ldots, \omega_{1,N}),\ \omega_{1,i} = \ \frac{1}{N}. \tag{1}$$

Step 2. Perform M iterations $(m = 1, 2, \ldots, M)$ as follows.

(i) Use the training set with the weight distribution D_m to learn a basic classifier:

$$G_m(x) : X \to Y. \tag{2}$$

(ii) Calculate the classification error rate of $G_m(x)$ on the training set,

$$e_m = \sum_{i=1}^{N} \omega_{m,i} I(G_m(x_i) \neq y_i). \tag{3}$$

From this formula, we know that the error rate e_m of $G_m(x)$ on the training set is the sum of the weights of the misclassified samples by $G_m(x)$.

(iii) Calculate the coefficient α_m of $G_m(x)$ where α_m indicates the importance of $G_m(x)$ in the final classifier. The smaller the classification error rate, the greater the role of the basic classifier in the final classifier.

$$\alpha_m = \ \frac{1}{2} \log \frac{1 - e_m}{e_m}. \tag{4}$$

(iv) Update the weight distribution of the training set and then get a new weight distribution for the next iteration:

$$D_{m+1} = \left(\omega_{m+1,1}, \omega_{m+1,2}, \ldots, \omega_{m+1,N}\right), \tag{5}$$

$$\omega_{m+1,i} = \frac{\omega_{m,i}}{Z_m} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right), \tag{6}$$

$$Z_m = \sum_{i=1}^{N} w_{m,i} \exp(-\alpha_m y_i G_m(x_i)). \tag{7}$$

Z_m is the normalization factor that makes D_{m+1} become a probability distribution. AdaBoost can focus on those samples that are more difficult to distinguish.

Step 3. Combine all weak classifiers to get the final classifier:

$$G\left(x\right) = \sum_{m=1}^{M} \alpha_m G_m(x). \tag{8}$$

Obviously, if some samples are misclassified, their weights will be increased in the next iteration. The weights of the correctly classified samples will be reduced in the next iteration and the error rate e_m is continuously reduced.

2.2 Clustering analysis

Clustering is an important method of data mining and can be used to discover the overall distribution pattern of samples in the feature space. A high-dimensional dataset can be divided into a plurality of separated connected regions containing similar point sets according to the information of samples and their features. The goal of clustering is that samples in the same cluster have high similarity and the similarity of samples in different clusters is low.

The k-means method is a commonly used partition-based clustering algorithm that divides samples in a dataset into k clusters and makes the selected partitioning standard function reach the optimal solution. It has many advantages including a fast convergence rate, a great clustering effect and strong interpretability. Therefore, we use k-means to cluster our datasets in the experiments. The main idea of k-means algorithm [16] is shown as follows. First, select k initial centroids. Then, each sample point is assigned to the nearest centroid and all sample points assigned to the same centroid form a cluster. Finally, the centroid of each cluster is updated according to the point set of each cluster. It repeats the assignment and updates steps until each cluster does not change or the centroid does not change.

2.3 CAdaBoost algorithm

Figure 1 shows the overall flow of our CAdaBoost algorithm. A training set is first divided into k clusters by the clustering algorithm. Then, in every iteration of training a weak classifier, we first calculate a misclassification degree for every cluster and then decide if a misclassified sample of a cluster is a noisy point according to the misclassification degree of the cluster. We will flexibly update the weight of a misclassified sample according to its decision. Note that we still use the strategy of AdaBoost to update the weight of a correctly-classified sample.

Specifically, after each iteration, the number C_j of the misclassified samples in the j-th cluster is counted where $j = 1, 2, \ldots, k$, and the misclassification degree of the j-th cluster is calculated as follows:

$$R_j = \frac{C_j}{\sum_{t=1}^{k} C_t}. \tag{9}$$

In view of the misclassification degree of a cluster, the cluster is considered as one of the following three cases.

(1) If the misclassification degree of a cluster is very low, we believe that the current classifier learns the characteristics of the samples in the cluster sufficiently and a misclassified sample in the cluster has a relatively high probability of being a noisy point. Therefore, we should limit the magnitude of increasing the weight of such a misclassified sample in order to reduce its impact on the classifier in the next round of training.

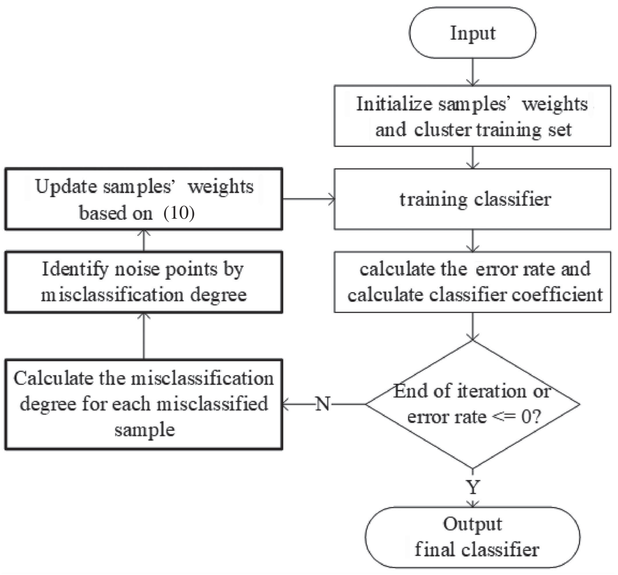
Input
Initialize samples' weights and cluster training set
training classifier
calculate the error rate and calculate classifier coefficient
End of iteration or error rate <= 0?
Y
Output final classifier
N
Calculate the misclassification degree for each misclassified sample
Identify noise points by misclassification degree
Update samples' weights based on (10)

Figure 1 The flow of the CAdaBoost.

(2) If the misclassification degree of a cluster is high, it means that the current classifier does not adequately extract the sample features in the cluster. Therefore, the misclassified samples of this cluster should be paid more attention in the next round of training, i.e., the magnitude of increasing the weights of the misclassified samples in such a cluster should be relatively high.

(3) The third case is that the misclassification degree of a cluster is between the above two cases. In other words, the magnitude of increasing the weight of a misclassified sample in such a cluster should be between the above two cases.

Based on the above idea, we use the following formula to flexibly adjust the weight of the misclassified sample x_i in the cluster D_j in the $(m+1)$-th iteration:

$$\omega'_{m+1,i} = \begin{cases} \omega_{m+1,i}, & \theta_2 \leqslant R_j, \\ c \times \omega_{m+1,i}, & \theta_1 \leqslant R_j < \theta_2, \\ \dfrac{1}{N}, & R_j < \theta_1, \end{cases} \tag{10}$$

where θ_1 and $\theta_2 = \frac{1}{k}$ are two thresholds, $\theta_1 < \theta_2$, and $\frac{w_{m,i}}{w_{m+1,i}} \leqslant c < 1$. In practice, we choose appropriate θ_1 and k by cross-validation. When we use (10) to calculate the weight of a misclassified sample in the $(m+1)$-th iteration, we still need to calculate its weight by (6). When the misclassification degree of a cluster is less than θ_1, we believe that those misclassified samples in the cluster are probably outliers and their weights are reset to the initial value $\frac{1}{N}$. This can limit the spread of increasing their weights and then avoid a big impact on the training of the next classifier. When the misclassification degree of a cluster is greater than or equal to θ_2, we believe that the characteristics of the misclassified samples in this cluster are not fully obtained and thus their weights are set by a relative high value. Here we set them as the same as that in AdaBoost (i.e., $w'_{m+1,i} = w_{m+1,i}$). When the misclassification degree of a cluster is less than θ_2 but greater than or equal to θ_1, the weight of the misclassified samples in the cluster is assigned by a value that is greater than $\frac{1}{N}$ but less than $w_{m+1,i}$. Therefore, $w_{m+1,i}$ is multiplied by a coefficient c such that $\frac{w_{m,i}}{w_{m+1,i}} \leqslant c < 1$. Algorithm 1 shows CAdaBoost.

2.4 Rationality of CAdaBoost

We use $E = E_1 \cup E_2 \cup E_3$ to represent the set of misclassified samples, where E_1 represents the set of misclassified samples whose misclassification degree is greater than or equal to θ_2, E_2 represents the set of misclassified samples whose misclassification degree is less than θ_2 but greater than or equal to θ_1, and E_3 represents a set of misclassified samples whose misclassification degree is less than θ_1. We divide the misclassified samples into three cases: one (i.e., E_1) is that they are not thought of as noises, one (i.e.,

Algorithm 1 Clustering based AdaBoost

```
Input: dataset $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$, iteration count $M$, parameter $\theta_1$, $k$ and $c$.
Output: $G(x) = \text{sign}(\sum_{m=1}^{M} \alpha_m G_{m(x)})$
 1: Initialize: $\omega_{1,i} = \frac{1}{N}$, $i = 1, 2, \dots, N$;
 2: Divided $D$ into $k$ clusters by a clustering algorithm as $D_1, D_2, \dots, D_k$;
 3: while $m \neq M$ and $e_m > 0$ do
 4:     Training a basic classifier $G_m(x) : X \to Y$;
 5:     For each $j \in \{1, 2, \dots, k\}$, count the number of misclassified samples in the $j$-th cluster: $C_j$;
 6:     Calculate the misclassification degree $R_j$ by (9) for each cluster;
 7:     $m = m + 1$;
 8:     Calculate $e_m$ and $\alpha_m$ by (3) and (4), respectively;
 9:     Calculate the sample weights $\omega'_{m+1,i}$ by (10);
10:     $\omega'_{m+1} = (\omega'_{m+1,1}, \omega'_{m+1,2}, \dots, \omega'_{m+1,N})$;
11: end while
```

E_3) is that they are though of as noises, and one (i.e., E_2) is that we are not sure whether they are noises or not. According to (3) and (10), the formula of computing error rate becomes

$$e_m = \sum_{x_i \in E_1} \omega_{m,i} + \sum_{x_i \in E_2} c \cdot \omega_{m,i} + \sum_{x_i \in E_3} \frac{1}{N}. \tag{11}$$

Obviously when $E_2 = \emptyset$ and $E_3 = \emptyset$, our method degenerates into the traditional AdaBoost algorithm (i.e., Eq. (11) is the same with (3)). But E_2 and E_3 are rarely both empty according to our observations on experiments. Hence, when E_2 and/or E_3 are not empty, the error rate computed by (11) is less than the error rate computed by (3), i.e., Eq. (11) is more accurate than (3). Therefore, this guarantees that the next training and the obtained classifier are more accurate.

3 Experiments

In this section, we first recall three state-of-the-art improvements of AdaBoost, and then illustrate our comparison experiments.

3.1 Improvemed versions of AdaBoost

3.1.1 *AdaCost algorithm*

A classifier has two kinds of misclassifications: predicting positive samples as negative samples and predicting negative samples as positive samples. AdaCost algorithm [13] uses the cost of misclassification to update the distribution of training samples, which has different costs for the two misclassifications in order to reduce the losses caused by misclassification. In AdaCost,

$$\omega_{m+1,i} = \omega_{m,i} \cdot \mathrm{e}^{-\alpha_m y_i G_m(x_i)} \beta_{\text{sign}(G_{m(x_i)}, y_i)}, \tag{12}$$

$$\alpha_m = \frac{1}{2} \log \frac{1 + r_m}{1 - r_m}, \tag{13}$$

$$r_m = \sum_i \omega_{m,i} \cdot \mathrm{e}^{-\alpha_m y_i G_m(x_i)} \beta_{\text{sign}(G_{m(x_i)}, y_i)}, \tag{14}$$

where β is called as a cost adjustment function which is preset.

3.1.2 *LogitBoost algorithm*

LogitBoost algorithm [14] uses Newton steps for optimizing the logistic loss which is more robust than exponential loss. LogitBoost follows the below steps.

Step 1. Initialize the weight $\omega_i = \frac{1}{N}$, $F(x) = 0$ and probability estimates $p(x_i) = \frac{1}{2}$.

Step 2. Repeat for $m = 1, 2, \dots, M$.

(i) Compute the working response and weight:

$$z_i = \frac{y_i - p(x_i)}{p(x_i)(1 - p(x_i))}, \tag{15}$$

$$\omega_i = p(x_i)(1 - p(x_i)). \tag{16}$$

Table 1 Balanced datasets' information

Dataset	Number of positive instances	Number of negative instances	Total number of instances	Dim
Breast-cancer	357	212	569	30
Clean	207	269	476	166
Cmc	511	333	844	9
Cylinder-bands	178	99	277	39
Ionosphere	225	126	351	34
Spambase	1813	1788	4601	57
Waveform	1655	1653	3308	40

(ii) Fit the function $f_m(x)$ by a weighted least-squares regression of z_i to x_i using weights ω_i.
(iii) Update

$$F(x) = F(x) + \frac{1}{2} f_m(x), \tag{17}$$

$$p(x) = \frac{\mathrm{e}^{F(x)}}{\mathrm{e}^{F(x)} + \mathrm{e}^{-F(x)}}. \tag{18}$$

Step 3. Output the classifier:

$$\mathrm{sign}(F(x)) = \mathrm{sign}\left(\sum_{m=1}^{M} f_m(x)\right). \tag{19}$$

3.1.3 *SPLBoost algorithm*

SPLBoost [15] assigns latent weight v to the exponential losses of training samples to overcome the problem that the exponential loss is directly minimized and the outliers whose losses are usually very large are easily paid more attention to AdaBoost. Algorithm 2 shows the details of SPLBoost.

Algorithm 2 SPLBoost algorithm

```
Input: dataset $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, iteration count $M$, parameter $\lambda$.
Output: Classifier $\mathrm{sign}(\sum_{m=1}^{M} \alpha_m G_m(x))$.
1: Initialize: $\omega_i = \frac{1}{N}, v_i = 1, i = 1, 2, \ldots, N$;
2: for $m = 1$ to $M$ do
3:     while $n$ not converge do
4:         Fit classifier $f_m(x) \to Y$ using weights $\frac{v_i \omega_i}{\sum_i v_i \omega_i}$ on the training data;
5:         Compute $\mathrm{err} = \sum_{y_i \neq f(x_i)} \frac{v_i \omega_i}{\sum_i v_i \omega_i}$ and $\alpha_m = \frac{1}{2} \log \frac{1-\mathrm{err}}{\mathrm{err}}$;
6:         Compute $v$;
7:     end while
8:     Set $\omega_i = \omega_i \mathrm{e}^{\log \frac{1-\mathrm{err}}{\mathrm{err}}}$;
9: end for
```

3.2 Experimental results and analysis

We validate the effectiveness of our algorithm on 22 public datasets. C4.5 is a decision tree algorithm and has the characteristics of flexibility and fast training. Therefore, just like [15], we choose C4.5 as the base classifier in our comparison experiments.

The first experiment is performed on seven datasets from UCI1) that have a relatively balanced distribution of positive and negative samples. We use the traditional accuracy as the evaluation criteria. Table 1 describes the basic information of the seven datasets and Table 2 shows the experimental results. It can be seen that the CAdaBoost algorithm achieves the best accuracy.

To further validate the advantage of our algorithm, the second group of experiments is performed on thirteen unbalanced datasets from the KEEL2) and two larger datasets from Library3). We use the AUC value as the evaluation criteria because accuracy cannot evaluate the performance of a method on an imbalanced dataset and AUC is often used for imbalanced datasets. Table 3 shows the basic information

1) http://archive.ics.uci.edu.
2) http://www.keel.es.
3) http://odds.cs.stonybrook.edu.

Table 2 Accuracy values on balanced datasets

Dataset	CAdaBoost	AdaBoost	AdaCost	LogitBoost	SPLBoost
Breast-cancer	**0.9770**	0.9510	0.9580	0.9603	0.9613
Clean	**0.8319**	0.7983	0.7863	0.8081	0.8016
Cmc	**0.6872**	0.6777	0.6825	0.6405	0.6755
Cylinder-bands	**0.7714**	0.6429	0.6571	0.6665	0.6786
Ionosphere	**0.9432**	0.9091	0.9205	0.9205	0.9137
Spambase	**0.9407**	0.9235	0.9270	0.9262	0.9255
Waveform	**0.9335**	0.9154	0.9262	0.8972	0.9274

Table 3 Imbalanced datasets' information

Dataset	Number of positive instances	Number of negative instances	Total number of instances	Dim
Abalone19	32	4142	4174	8
Ecoli4	20	316	336	7
Galss0	76	138	214	9
Glass2	13	201	214	9
Glass4	9	205	214	9
Glass5	29	185	214	9
Iris0	35	180	215	4
New-thyroid	35	180	215	5
Pima	329	1979	3308	8
Yeast1458vs7	51	463	514	8
Yeast2vs4	20	462	482	8
Yeast5	35	1449	1484	8
Yeast6	35	1449	1484	8
Smtp	30	95126	95156	5
Forestcover	2747	283301	286048	6

Table 4 AUC values on imbalanced datasets

Dataset	CAdaBoost	AdaBoost	AdaCost	LogitBoost	SPLBoost
Abalone19	**0.7500**	0.7425	0.7325	0.7047	**0.7500**
Ecoli4	0.8212	0.7671	0.7945	**0.8307**	0.8026
Galss0	**0.7749**	0.7542	0.7375	0.7542	0.7417
Galss2	**0.8407**	0.8137	0.8250	0.7454	0.8166
Galss4	**1.0000**	0.9245	0.9623	**1.0000**	**1.0000**
Galss5	**0.9386**	0.9333	0.9375	0.8905	0.9289
Iris0	**1.0000**	0.9878	0.9878	**1.0000**	**1.0000**
New-thyroid	**0.9822**	0.9444	0.9400	0.9662	0.9488
Pima	**0.9923**	0.9845	0.9900	0.9804	0.9890
Yeast1458vs7	**0.8996**	0.8832	0.8790	0.8916	0.8791
Yeast2vs4	**0.8871**	0.8571	0.8528	0.8706	0.8706
Yeast5	**0.9098**	0.8862	0.9025	0.8958	0.8403
Yeast6	**0.8114**	0.7857	0.7808	0.7445	0.7843
Smtp	**0.8571**	0.8333	0.8500	0.8500	**0.8571**
Forestcover	**0.9705**	0.9561	0.9577	0.9456	0.9652

of these imbalanced datasets and Table 4 shows the experimental results. Obviously, CAdaBoost also achieves better performance.

Figures 2 and 3 show the variation of the accuracy/AUC scores of CAdaBoost on each dataset when adjusting the number of clusters k and the threshold θ_1. In the experiment, $c = \frac{1}{2} \cdot (1 + \frac{w_{m,i}}{w_{m+1,i}})$, the value range of k is $\{3, 4, 5, 6\}$ and the threshold θ_1 range is $\{0.01, 0.02, \ldots, 0.09\}$. We can see that the impact of θ_1 and k is different to different datasets. θ_1 and k change frequently on the breast-cancer, ionoshere, glass5 and other datasets that make the experimental results have large fluctuations, but they change little on clean, cmc, ecoli4 and other datasets that do not make the experimental results have big fluctuations.

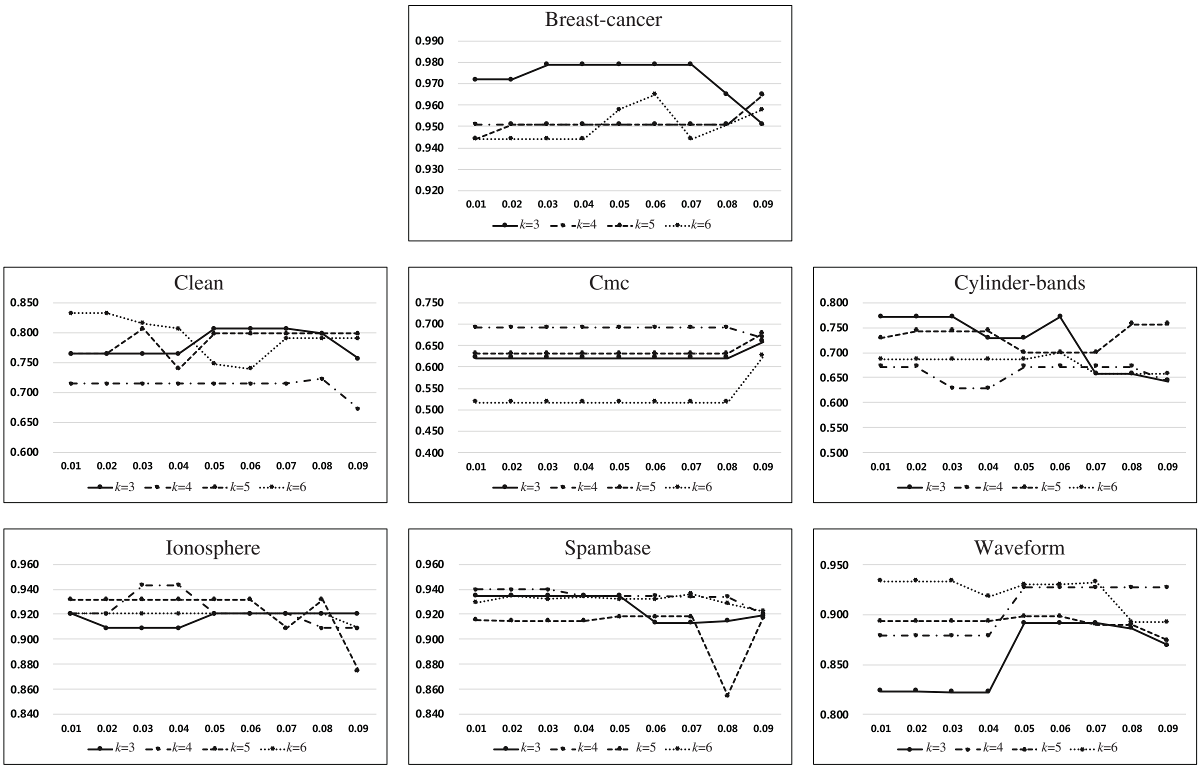

Figure 2 The change curves of accuracy values of CAdaBoost on balanced datasets with different k and θ_1.

Table 5 Training set and test set

	Training set	Test set
1	April data	May data
2	May data	June data
3	April and May data	June data

Table 6 Information of our transaction datasets

	Number of fraudulent transactions	Number of legal transactions	Total number of transactions	Dim
2017.04	13271	1229356	1242627	46
2017.05	27122	1188722	1215844	46
2017.06	24898	1017294	1042192	46

4 Application of CAdaBoost on transaction fraud detection

We apply our method in the system of detecting transaction fraud. Here we illustrate some experimental results on a big dataset from a financial company of China. It contains more than 3.5 million transaction records of B2C in which there are about 65300 fraud ones. Obviously, the dataset is imbalanced seriously.

Random forest [17, 18] is a classic ensemble learning method, which is often used in transaction fraud detection. Therefore, we also compare CAdaBoost with random forest in these experiments. Tables 5 and 6 show the information of dataset, training sets and test sets. Table 7 shows the AUC scores of the six algorithms. Obviously, our CAdaBoost achieves the best result.

We also use F_1, recall and precision to evaluate our method, as shown in Tables 8–10. For the two-category problem, samples can be divided into true positive examples (TP), false positive examples (FP), true negative examples (TN) and false negative examples (FN) according to the combination of their real category and the machine learning prediction category. $F_1 = \frac{2\times P\times R}{P\ +\ R}$ is the harmonic mean of the precision rate $P = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$ and the recall rate $R = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$. F_1, just like AUC, is also an important measure for the performance of a machine learning method on imbalanced datasets. CAdaBoost obtains the best

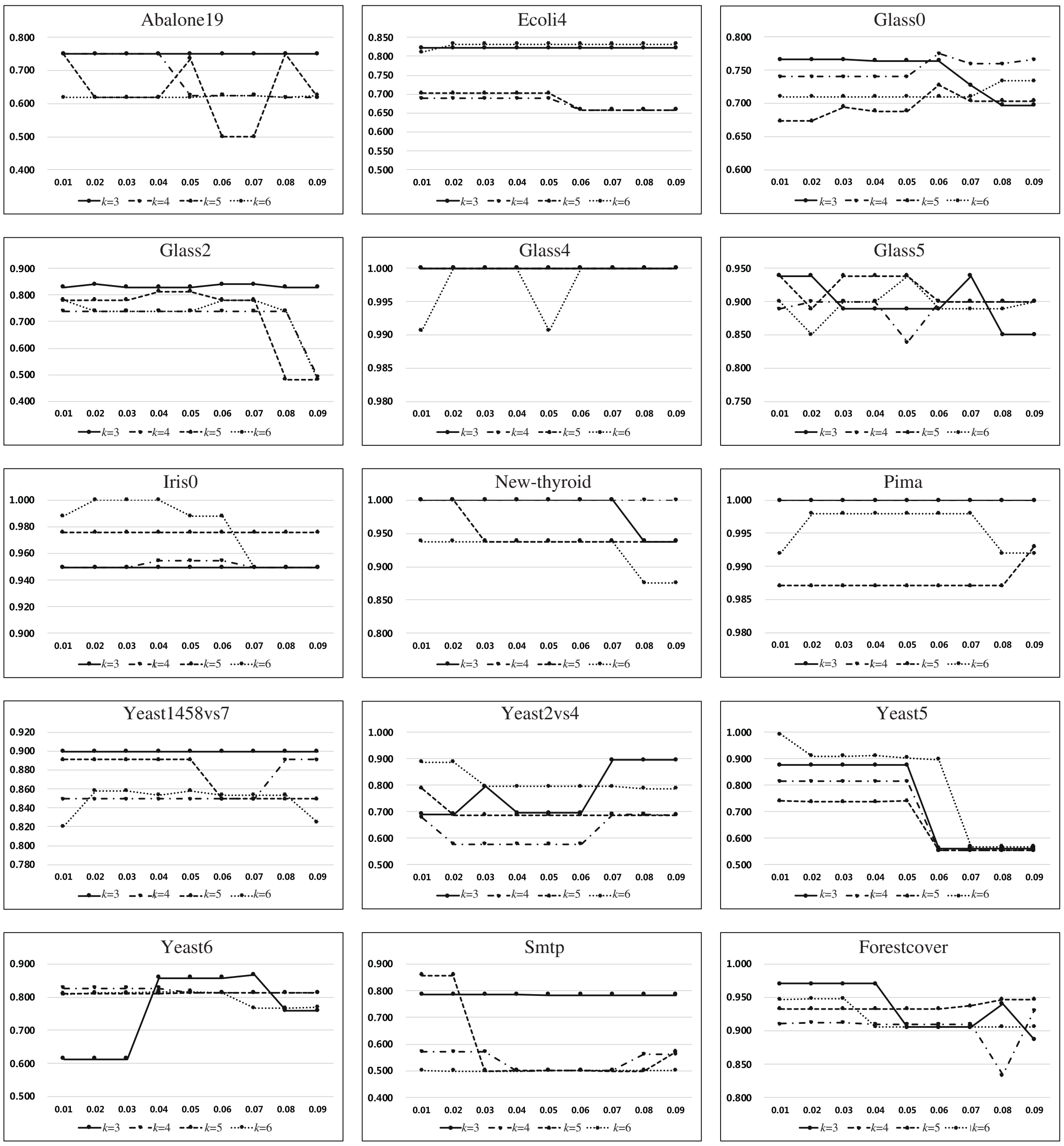

Figure 3 The change curves of AUC values of CAdaBoost on imbalanced datasets with k and θ_1.

Table 7 AUC scores on transaction datasets

	CAdaBoost	AdaBoost	AdaCost	LogitBoost	SPLBoost	RandomForest
1	**0.9446**	0.8997	0.6649	0.9003	0.9027	0.8652
2	**0.9481**	0.9139	0.9129	0.9225	0.9187	0.8809
3	**0.9432**	0.9133	0.9074	0.8961	0.9149	0.8769

F_1 scores. From the recall scores in Table 9, we can see that CAdaBoost can detect more fraud samples (although its precision is a little lower than others, i.e., a little more legal transactions are intercepted as fraud ones). Random forests achieve the best results in precision scores, but the coverage of fraud samples is low compared to others. Financial companies usually hope to intercept more fraud transactions (i.e., a high recall). At the same time, we find that transaction data have an obvious clustering phenomenon in the feature space and thus our CAdaBoost can be well applied to the transaction fraud detection.

Table 8 F1 scores on transaction datasets

	CAdaBoost	AdaBoost	AdaCost	LogitBoost	SPLBoost	RandomForest
1	**0.8327**	0.8052	0.7641	0.7997	0.8085	0.7752
2	**0.7727**	0.7561	0.7641	0.7600	0.7566	0.7543
3	**0.7694**	0.7508	0.7609	0.7429	0.7621	0.7522

Table 9 Recall scores on transaction datasets

	CAdaBoost	AdaBoost	AdaCost	LogitBoost	SPLBoost	RandomForest
1	**0.8866**	0.8037	0.8344	0.8054	0.8097	0.8215
2	**0.8812**	0.8363	0.8344	0.8463	0.8471	0.7685
3	**0.8988**	0.8353	0.8231	0.8009	0.8387	0.7602

Table 10 Precision scores on transaction datasets

	CAdaBoost	AdaBoost	AdaCost	LogitBoost	SPLBoost	RandomForest
1	0.7951	0.8068	0.7047	0.7940	0.8074	**0.8215**
2	0.6860	0.6968	0.7047	0.6945	0.6836	**0.7406**
3	0.6726	0.6886	0.7074	0.6927	0.6983	**0.7443**

Notice that machine-learning-based transaction fraud detection is a complex engineering that not only requires a good classification method but also needs to deal with the feature mining [19], data imbalance [20], and concept drift [21]. Here we focus on the improvement of the classification method. More importantly, our improvement is not complex but its effect is very obvious.

5 Conclusion

CAdaBoost adapts a clustering-based method to dynamically predict those noisy points and flexibly adjust the weights of misclassified samples. It obtains a good performance especially when a dataset has an obvious clustering phenomenon. CAdaBoost can achieve a good result when we apply it to the transaction fraud detection. In the future, we plan to consider the distribution of misclassified samples in more detail and improve our method furthermore.

Acknowledgements This work was supported in part by National Key Research and Development Program of China (Grant No. 2018YFB2100801), and Fundamental Research Funds for Central Universities of China (Grant No. 22120190198). Authors would like to thank anonymous reviewers for their constructive comments.

References

1 Yu B, Xu Z B. A comparative study for content-based dynamic spam classification using four machine learning algorithms. Know-Based Syst, 2008, 24: 355–362

2 Ju W H, Vardi Y. A hybrid high-order Markov chain model for computer intrusion detection. J Comput Graph Stat, 2001, 10: 277–295

3 Shen L, Bai L, Bardsley D, et al. Gabor feature selection for face recognition using improved adaboost learning. In: Li S Z, Sun Z, Tan T, eds. Advances in Biometric Person Authentication. Berlin: Springer, 2005. 3781: 39–49

4 Panigrahi S, Kundu A, Sural S, et al. Credit card fraud detection: a fusion approach using Dempster-Shafer theory and Bayesian learning. Inf Fusion, 2009, 10: 354–363

5 Salzberg S L. C4.5: programs for machine learning. Mach Learn, 1994, 16: 235–240

6 Cortes C, Vapnik V. Support vector network. Mach Learn, 1995, 20: 273–297

7 Ng A Y, Jordan M I. On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, Cambridge, MIT Press, 2001. 841–848

8 Zhou Z H. Ensemble Learning. In: Encyclopedia of Biometrics. Boston: Springer, 2009

9 Dietterich T G. Ensemble methods in machine learning. In: Proceedings of International Workshgp on Multiple Classifier Systems, 2000. 1857: 1–15

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

11 Breiman L. Bagging predictors. Mach Learn, 1996, 24: 123–140

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

13 Wei F, Stolfo S J, Zhang J X, et al. AdaCost: misclassification cost-sensitive boosting. In: Proceedings of International Conference on Machine Learning (ICML-99), Bled, 1999. 97–105

14 Friedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). Ann Statist, 2000, 28: 337–407

15 Wang K, Wang Y, Zhao Q, et al. SPLBoost: an improved robust boosting algorithm based on self-paced learning. 2017. ArXiv: 1706.06341

16 Wong J A, Hartiganm A. Algorithm AS 136: a k-means clustering algorithm. J R Stat Soc, 1979, 28: 100–108

17 Breiman L. Random forests. Mach Learn, 2001, 45: 5–32

18 Xuan S Y, Liu G J, Li Z C, et al. Random forest for credit card fraud detection. In: Proceedings of IEEE 15th International Conference on Networking, Sensing and Control (ICNSN), Zhuhai, 2018. 27–29

19 Jiang C, Song J, Liu G, et al. Credit card fraud detection: a novel approach using aggregation strategy and feedback mechanism. IEEE Internet Things J, 2018, 5: 3637–3647

20 Zhang F J, Liu G J, Li Z C, et al. GMM-based undersampling and its application for credit card fraud detection. In: Proceedings of the 32nd International Joint Conference on Neural Network (IJCNN2019), Budapest, 2019. 14–19

21 Zheng L, Liu G, Yan C, et al. Transaction fraud detection based on total order relation and behavior diversity. IEEE Trans Comput Soc Syst, 2018, 5: 796–806