Neurocomputing 508 (2022) 153–169

Contents lists available at ScienceDirect

Neurocomputing

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

Improved AdaBoost algorithm using misclassified samples oriented feature selection and weighted non-negative matrix factorization

Youwei Wang [a], Lizhou Feng [b,*], Jianming Zhu [a], Yang Li [a], Fu Chen [a]

[a] School of Information, Central University of Finance and Economics, Beijing 100081, China
[b] School of Statistics, Tianjin University of Finance and Economics, Tianjin 300222, China

ARTICLE INFO

Article history:
Received 17 August 2021
Revised 28 July 2022
Accepted 3 August 2022
Available online 12 August 2022

Keywords:
Classification performance
Adaptive boosting
Weighted non-negative matrix factorization
Misclassification degrees
Feature selection

ABSTRACT

To improve the classification performance of existing adaptive boosting (AdaBoost) based algorithms effectively, an improved AdaBoost algorithm based on misclassified samples oriented feature selection and weighted non-negative matrix factorization (WNMF) is proposed in this paper. Firstly, in order to consider the effects of sample weights, a misclassified samples oriented feature selection (called MOFS) is proposed to select the most discriminative features which occur in the samples with high weights. Secondly, the explicit features and the part-based features of the training samples are both considered, and the WNMF algorithm is introduced and combined with MOFS to reduce the dimension of the training sample set. Finally, the concept of misclassification degree is introduced and a fine grained sample weight updating method is proposed to distinguish the samples with different misclassification degrees. Numerical experiments show that the proposed MOFS method achieves higher accuracy compared to traditional feature selection methods, and the proposed MOFS and WNMF based AdaBoost method obtains significant improvement on classification accuracy when comparing with typical existing AdaBoost based algorithms using different classifiers.

1. Introduction

Bagging and boosting are two types of ensemble learning algorithms which are widely used in the areas of data classification and pattern recognition [1,2]. In order to improve the classification accuracy, a bagging algorithm generates multiple versions of a predictor using bootstrap and then uses them to get an aggregated predictor. There are numerous bagging algorithms, such as random forest [3], EUSBagging [4], RB Bagging [5], etc. Different with the bagging based algorithms, most boosting algorithms iteratively train a weak classifier and combine the results of different weak classifiers to form a final strong classifier. The typical examples of boosting algorithms include Boosting [6], Adaptive Boosting (AdaBoost) [7–9], Gradient Boosting Decision Tree (GBDT) [10], XGBoost [11], etc. Compared to boosting algorithms, bagging algorithms are more effective for the advantage of parallelism. However, as boosting algorithms fully consider the importance of sample weights and can improve the classification ability effectively, they have been used widely in many fields like text classification, image classification and intrusion detection [4].

The basic idea of AdaBoost is to increase the weights of misclassified samples and reduce the weights of correctly classified samples. When combining multiple weak classifiers, a weighted majority voting method is adopted to increase the weights of classifiers with low classification error rates and decrease the weights of classifiers with high error rates. Since AdaBoost is designed for solving two-class classification problems, many algorithms like AdaBoost.M1 [12], AdaBoost.M2 [12], AdaBoost.MH [13], SAMME [14], SAMMER [15], SAMME.R[16] and AdaBoost.RT [17] are proposed to extend AdaBoost to multi-class classification or regression cases. In recent years, people studied from the aspects of improving the robustness and classification accuracy, and proposed many typical AdaBoost based algorithms, such as: Rob_MulAda [18], MF_ADA [19], M_ADA_A [20], M_ADA_A_SMV [21], M_ADA_A_PSO [21], BPSO-Adaboost-KNN [22], etc.

Based on the existing AdaBoost based algorithms, we propose an improved AdaBoost algorithm using misclassified samples oriented feature selection and weighted non-negative matrix factorization (WNMF) in this paper. The rest of this paper is organized as follows. Section 2 introduces the related work of AdaBoost algorithms. Section 3 gives the details of WNMF algorithm and SAMME algorithm. Section 4 gives the implements of the proposed

* Corresponding author.

https://doi.org/10.1016/j.neucom.2022.08.015
0925-2312/

algorithm. Section 5 gives the experimental results and analysis. Section 6 concludes the whole paper.

2. Related work

Currently, the research on AdaBoost mainly focuses on the aspects of noise detection, sampling and classifier enhancing. Noise detection identifies and removes the noisy samples to avoid the effect of noisy samples on classification performance [18]. As a traditional AdaBoost algorithm can hardly represent the distribution of the minority because of the domination of the majority, sampling methods are often used to select the most representative samples from the training set to solve the dataset imbalance problem [23]. With respect to classifier enhancing, methods like feature selection, loss function optimization and weak classifier combination are often used to improve the classification ability of AdaBoost. Details are given as follows:

(1) Noise detection methods

In many practical applications of AdaBoost algorithms, the collected training set often contains some noisy samples (noises), which are detrimental to the classifiers' generalization performance. The traditional AdaBoost algorithms assign larger weights to misclassified samples and ignore the mislabeled noises, deducing that both the misclassified non-noisy samples and the misclassified noisy samples are assigned larger weights. To enhance the noise tolerance of AdaBoost, Muhlenbach proposed an approach that detects noisy samples based on the class label comparison between each example and its k nearest neighbors in the training set [24]. However, this approach shrinks the training set size and in some cases degrades the generalization performance of the learned classifiers especially when the training set size is small. Servedio designed a boosting algorithm that generates smooth weight distributions on the training set and prevents from assigning too large weight to any single training sample [25]. Although this approach tries to reduce the harmful effect of noises, it does not explicitly define which and how many examples in the original training set should be considered as mislabeled noises. For solving this problem, Cao proposed a noise detection based AdaBoost algorithm (ND_AdaBoost) which first divides all the training samples into two categories: the non-noisy samples and the mislabeled noisy sampes, and then assigns different weights to the samples according to their categories [26]. Experimental results demonstrate that ND_AdaBoost is more robust than AdaBoost on training sets with mislabeled noises. Further, Sun et al. proposed a new AdaBoost algorithm based on Cao's algorithm [18]. This algorithm sets the noisy samples' weights to zero if they are misclassified, avoiding the unlimited increase of noisy samples' weights during the iteration process.

(2) Sampling methods

In real classification tasks, data imbalance occurs when the sample numbers of two classes are far from equal, deducing that the samples in the class with fewer samples are often under represented. In recent years, data imbalance has been widely regarded as a very challenging problem in data mining [27]. Among existing sampling techniques, oversampling and undersampling are the most representative methods which are widely used to provide a balanced distribution. Oversampling aims to increase the sample number of the class with fewer samples (minority class) to balance the sizes of two classes [28]. Typical oversampling methods include SMOTE [29], borderline-SMOTE [30], ADASYN [31], boundary oversampling (BOS) [32], KernelADASYN [33] and GIREnOS [23]. Undersampling attempts to balance the sizes of different classes by reducing the sample number of the class with more samples (majority class). The easiest undersampling method is random undersampling (RUS), which extracts samples randomly from the majority classes. EUS [34] is a relatively new undersampling method which uses the evolutionary mechanism to obtain a best majority class subset according to a predefined fitness function and is shown to achieve the best classification performance by combining the decision tree classifier [4]. Similar to oversampling that generates boundary positive samples, Bao's method uses undersampling to reserve borderline negative instances [35]. This method combines undersampling technique, real AdaBoost, cost-sensitive weight modification and adaptive boundary decision strategy to build a hybrid algorithm, and is proven to be a promising alternative that can be applied to various imbalanced classification domains.

(3) Classifier enhancing methods

As the weak classifier training process is time consuming when dealing with datasets with high dimensionalities [36], feature selections are mainly used to improve the execution speed of AdaBoost algorithms. Feature selections mainly include two types: wrappers and filters [37]. The main idea of a wrapper is to evaluate the performance of a feature subset by utilizing a classification algorithm. Different with wrappers, filters like the Information Gain (IG) [38], Improved Comprehensive Measurement Feature Selection (CMFSX) [39], Improved Gini Index (IMGI) [40] and Chi-squared Test (CHI) [41] use specific metrics to evaluate the category differentiating ability of the features, deducing lower execution time than wrappers. Yao et al. applied Particle Swarm Optimization (PSO) algorithm to optimize the features' distribution according to their performance, and then obtained the best feature subset in each iteration [8]. Guo et al. employed Binary Particle Swarm Optimization (BPSO) to search the best feature subset when Area Under Curve (AUC) measurement is used as the fitness function [22]. Mazini used Artificial Bee Colony (ABC) algorithm to select the best features for network-based intrusion detection system, increasing the classification accuracy and the running speed effectively [42]. For improving the robustness of AdaBoost, Zhang combined the Principal Component Analysis (PCA) algorithm and the autoencoder neural network to select features from gene expression profiles [43].

Besides the feature selections, numerous methods are studied in the aspects like loss function optimization and sample weight updating to enhance the classification performance of AdaBoost algorithms. Cao et al. proposed a new loss function which prevents the sample weights from unrestricted increasing, avoiding the effects of noises or samples with singular values [44]. In order to capture the effects of imbalanced datasets, Zhou et al. defined the loss function as the product of the classification error rate and the AUC value [22]. Lee et al. proposed a new weight updating factor and applied it to weighted Support Vector Machine (SVM) which is used as the weak classifier of the AdaBoost algorithm [45]. As samples may be misclassified for many times, Tang et al. updated the sample weights by reducing the difference between the sample weights in the previous iteration and those in the current iteration to alleviate the imbalance of the sample weight distribution [19].

To further improve the classification performance, many approaches combining multiple different classifiers into an ensemble have been studied. Xiao et al. proposed an improved AdaBoost algorithm by applying clustering algorithm to the traditional AdaBoost [46]. This algorithm divides the dataset into several clusters and combines the weak classifiers trained on the clusters to form a strong classifier. Yang et al. used the Convolutional Neural

Networks (CNN) as the weak classifier, and evaluated the weights of the weak classifiers in a pre-training stage [47]. Based on Yang's algorithm, Yousefi et al. used multiple classifiers like Adaptive Network-based Fuzzy Inference System (ANFIS), Feed Forward Neural Network (FFNN) and the Recurrent Neural Network (RNN) as weak classifiers [48], improving the robustness and classification accuracy effectively. Gao et al. deployed an adaptive ensemble learning technique which combines several classifiers, such as Decision Tree (DT), Random Forest (RF), KNN and Deep Neural Network (DNN) [49]. Zhou et al. combined multiple expert classifiers which incorporate the AUC metric into the boosting process for addressing the class imbalance problems [21]. Chen et al. used SVM, C4.5 decision tree and Artificial Neural Network (ANN) as the weak classifiers of AdaBoost, achieving higher classification accuracy than just using one of them [50].

However, existing AdaBoost based algorithms still face the following problems: (1) Traditional feature selections of filters (like IG [38], CMFSX [39], IMGI [40] and CHI [41]) are often used to reduce the feature dimensions of the training samples. However, most of these methods do not consider the effect of sample weights, deducing that the features which can distinguish the samples with high weights are not highlighted. (2) Most traditional AdaBoost algorithms like Adaboost.M1 and SAMME only consider the explicit features, ignoring the effect of part-based features which are very helpful in improving the accuracy of data classification tasks [51,52]. (3) Most traditional sample weight updating methods treat all misclassified samples equally, ignoring the fact that the samples with high misclassification degrees should draw more attention than the ones with low misclassification degrees during the iterations of AdaBoost algorithms.

For solving the above problems, this paper proposes a new AdaBoost algorithm based on misclassified samples oriented feature selection and weighted non-negative matrix factorization. The main contributions of this paper include: (1) we consider the dynamic changes of sample weights in each iteration of AdaBoost, and propose a misclassified samples oriented feature selection (called MOFS) to improve the classification ability of AdaBoost on dealing with misclassified samples; (2) we utilize the advantage of WNMF algorithm on both considering the data uncertainty and detecting the part-based features of the training samples, and combine the explicit features and the part-based features to improve the classification performance by using MOFS and WNMF; (3) we introduce the concept of misclassification degree and propose a fine grained sample weight updating method to distinguish the misclassified samples of which the misclassification degrees are often different.

3. Theory background

3.1. Weight non-negative matrix factorization (WNMF)

WNMF derives from non-negative matrix factorization (NMF) which transforms the original data into different components where the values are all no lower than zero. NMF was introduced as a dimension reduction method for pattern analysis. When a matrix with non-negative elements is given, NMF seeks to find a lower rank approximation of the data matrix where the factors that give the lower rank approximation are also non-negative [53]. Given the non-negative matrix $\boldsymbol{S} = [\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_i, ..., \boldsymbol{s}_M] \in \mathrm{R}^{M\times N}$, where M and N are the numbers of samples and features respectively, NMF can be used to decompose the transpose of $\mathbf{S}$ (denoted as $\mathbf{S}^{\mathrm{T}} \in \mathrm{R}^{N\times M}$) into the product of a non-negative matrix $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_i, \ldots, \mathbf{w}_N] \in \mathrm{R}^{N\times r}$ and another non-negative matrix $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_i, ..., \boldsymbol{h}_r] \in \mathrm{R}^{r\times M}$, such that [54,55]:

$$\begin{aligned} &\boldsymbol{S}^{\mathrm{T}} \approx \boldsymbol{WH} \\ &s.t. \mathbf{W} \geqslant 0, \mathbf{H} \geqslant 0 \end{aligned} \tag{1}$$

where $\mathbf{W}$ and $\mathbf{H}$ are the basis matrix and the weight matrix, respectively. The choice of r which is the dimension of the embedding space has no accurate way. In practice, r usually satisfies $(N + M) \times r < N \times M$, and its value is decided by experiments and experience [54]. For the solution of formula (1), the objective function based on Frobenius norm is written as:

$$\begin{aligned} &\min \Theta(\mathbf{W}, \mathbf{H}) = \| \mathbf{S}^{\mathrm{T}} - \mathbf{WH} \|_{\mathrm{F}}^2 = \sum_{\mathrm{i}=1}^{\mathrm{N}} \sum_{\mathrm{j}=1}^{\mathrm{M}} \left(\mathbf{S}_{\mathrm{ij}}^{\mathrm{T}} - \sum_{\mathrm{l}=1}^{\mathrm{r}} \mathbf{W}_{\mathrm{il}} \mathbf{H}_{\mathrm{lj}} \right) \\ &s.t. \mathbf{W} \geqslant 0, \mathbf{H} \geqslant 0 \end{aligned} \tag{2}$$

where $\| \boldsymbol{x} \|_{\mathrm{F}}^2$ denotes the Frobenius norm of matrix $\times$,$\mathbf{x}_{ij}$ denotes the value of the i_{th} row and j_{th} column in $\mathbf{x}$. The classical multiplicative iterative rules are defined as follows when using the gradient descent method for the above optimization problem [52]:

$$\mathbf{H} \leftarrow \mathbf{H} \odot \frac{\mathbf{W}^{\mathrm{T}} \mathbf{S}^{\mathrm{T}}}{\mathbf{W}^{\mathrm{T}} (\mathbf{WH})}, \mathbf{W} \leftarrow \mathbf{W} \odot \frac{\mathbf{S}^{\mathrm{T}} \mathbf{H}^{\mathrm{T}}}{(\mathbf{WH}) \mathbf{H}^{\mathrm{T}}} \tag{3}$$

where $\odot$ is the element-wise multiplication operation. The iteration stops when $\Theta(\boldsymbol{W}, \boldsymbol{H})$ is lower than a predefined threshold or after a certain number of iterations.

As NMF treats all elements in $\mathbf{S}$ equally, ignoring the effect of sample weights when defining the cost functions, Vincent et al. proposed the WNMF which can deal with uncertainty in the non-negative matrix $\mathbf{S}$. Given a non-negative weighting matrix $\mathbf{P} \in \mathrm{R}^{M\times N}$, the cost function $\Theta(\boldsymbol{W}, \boldsymbol{H})$ and the weighted multiplicative iterative rules can be defined as follows [52]:

$$\begin{aligned} &\min \Theta(\mathbf{W}, \mathbf{H}) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \sum_{\mathrm{j}=1}^{\mathrm{M}} \mathbf{P}_{\mathrm{ij}}^{\mathrm{T}} \left(\mathbf{S}_{\mathrm{ij}}^{\mathrm{T}} - \sum_{\mathrm{l}=1}^{\mathrm{r}} \mathbf{W}_{\mathrm{il}} \mathbf{H}_{\mathrm{lj}} \right) \\ &s.t. \mathbf{W} \geqslant 0, \mathbf{H} \geqslant 0 \end{aligned} \tag{4}$$

$$\boldsymbol{H} \leftarrow \boldsymbol{H} \odot \frac{\boldsymbol{W}^{\mathrm{T}} \left(\boldsymbol{P}^{\mathrm{T}} \odot \boldsymbol{S}^{\mathrm{T}} \right)}{\boldsymbol{W}^{\mathrm{T}} \left(\boldsymbol{P}^{\mathrm{T}} \odot \boldsymbol{WH} \right)}, \boldsymbol{W} \leftarrow \boldsymbol{W} \odot \frac{\left(\boldsymbol{P}^{\mathrm{T}} \odot \boldsymbol{S}^{\mathrm{T}} \right) \boldsymbol{H}^{\mathrm{T}}}{\left(\boldsymbol{P}^{\mathrm{T}} \odot \boldsymbol{WH} \right) \boldsymbol{H}^{\mathrm{T}}} \tag{5}$$

Obviously, we know from formula (5) that the time complexity of NMF and WNMF are both $O(ZMNr)$(Z is the number of iterations). Compared with NMF, WNMF can better emphasize certain samples or features by considering the data weight information, thus is widely used in text mining and document clustering.

3.2. Stagewise additive modeling using a multi-class exponential loss function (SAMME)

In order to solve the multi-classification problems, Zhu et al. proposed the SAMME method [14], which is the multi-class extension of the traditional two-class AdaBoost algorithm [12]. The details of SAMME are shown as follows:

Algorithm 1.SAMME.

Input: training sample matrix: $\mathbf{TS} = [\mathbf{s}_i] \in \mathrm{R}^{M\times N}(1 \le i \le M$, M and N are the numbers of samples and features in **TS**); label matrix of training samples: $\mathbf{Y} \in \mathrm{R}^{M\times 1}$; label set: $C=\{c_l\}$ $(1 \le l \le L$, L is the number of classes); number of iterations: Z; sample weight matrix: $\mathbf{D} = [\mathbf{D}_t] \in \mathrm{R}^{Z\times M}(1 \le t \le Z)$; weak classifier: h.

Output: the final classifier: H.

1: **for** i = 1:1:M
2: Initialize the sample weight $d_{1,i}$ in $\mathbf{D}_1$: $d_{1,i} = 1/M$.
3: **end for**
4: **for** t = 1:1:Z
5: Select a sub matrix $\mathbf{S}_t$ and its corresponding label matrix $\mathbf{Y}_t$ from **TS** and **Y**.
6: Obtain a trained classifier h_t:$h_t = \mathrm{TR}(h, \boldsymbol{S}_t, \boldsymbol{Y}_t)$, where TR $(h, \mathbf{S}_t, \mathbf{Y}_t)$ returns the trained classifier by applying h on $(\mathbf{S}_t, \mathbf{Y}_t)$.
7: Calculate the classification error rate ε_t:$\varepsilon_t = \sum_{i=1}^{M} d_i \mathrm{I}(h_t(s_i) \neq \boldsymbol{Y}(s_i))$, where $h_t(s_i)$ is the predicted label of sample s_i using classifier h_t, $\mathbf{Y}(s_i)$ is the real label of s_i, $\mathbf{I}(x)$ returns 1 if $\times$ is true, else returns 0.
8: Calculate the classifier weight of h_t:$\alpha_t = \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) + \ln(L-1)$.
9: **for** i = 1:1:M
10: Update s_i's sample weight:$d_{t+1,i} = \frac{d_{t,i} e^{\alpha_t \mathrm{I}(h_t(s_i) \neq \boldsymbol{Y}(s_i))}}{Z_t}$, where Z_t is the normalization factor and is set to:$Z_t = \sum_{i=1}^{M} d_{t,i} e^{\alpha_t \mathrm{I}(h_t(s_i) \neq \boldsymbol{Y}(s_i))}$.
11: **end for**
12: **end for**
13: Given a sample s, calculate s's final label H (s):$\mathrm{H}(s) = \arg\max_{c_k \in C} \sum_{t=1}^{Z} \alpha_t \mathrm{I}(h_t(s) == c_k)$.

It can be seen from Algorithm 1 that SAMME shares the same modular structure with traditional two-class AdaBoost algorithm and has a simple but subtle difference in step 8 which has the extra term $\ln(L-1)$. According to [14] we know that term $\ln(L-1)$ is not artificial and follows naturally from the multi-class generalization of the exponential loss in the binary case. Obviously, when L = 2, SAMME reduces to traditional two-class AdaBoost.

4. Implements of the proposed AdaBoost method

4.1. Misclassified samples oriented feature selection (MOFS) for AdaBoost

可以对mRMR引入权重改进，计算每个特征的得分

Most traditional AdaBoost algorithms just use typical feature selections of filters (like IG [38], CMFSX [39], IMGI [40] and CHI [41]) to reduce the data dimensions. Among these filters, CMFSX comprehensively considers the inter-category information and the intra-category information of a feature and can obtain the most discriminative features effectively. Given a feature $f_j(1 \le j \le N$, N is the number of features), CMFSX calculates the score of f_j as follows:

$$\begin{cases} \mathrm{CMFSX}(f_j) = \sum_{c_l} \mathrm{CMFSX}(f_j, c_l) = \sum_{c_l} \left(\frac{tfc_{jl}}{tc_l} \times \frac{tfc_{jl}}{tf_j} \times \frac{1}{p_l}\right) \\ p_l = \frac{tc_l}{M} \end{cases} \tag{6}$$

where CMFSX(f_j, c_l) is the CMFSX score of f_j with respect to category $c_l(1 \le l \le L$, L is the number of categories), tc_l is the number of samples in c_l, tfc_{jl} is the number of samples in which f_j occurs in c_l, tf_j is the number of samples which contain f_j, M is the number of samples in the entire training set, tfc_{jl}/tc_l is the intra-category score which denotes the category representative ability of f_j, tfc_{jl}/tf_j is the inter-category score [39] which denotes the category distinguishing ability of f_j.

However, as CMFSX ignores the sample weights when measuring the feature importance, it may lose some discriminative features which occur in the misclassified samples. For solving this problem, we propose a misclassified samples oriented feature selection (called MOFS) based on the CMFSX method. As the features' category distinguishing ability is affected by the feature distribution characteristics in different categories and has little relevance with the sample weights, MOFS retains the features' category distinguishing ability and calculates the inter-category scores using CMFSX method. However, MOFS considers the sample weights when calculating the intra-category score so as to pay more attention to the features contained in the samples with high weights. Given a feature f_j, the score of f_j is defined as follows:

$$\mathrm{MOFS}(f_j) = \sum_{c_l} \mathrm{MOFS}(f_j, c_l) = \sum_{c_l} \left(\frac{w_{jl}}{w_l} \times \frac{tfc_{jl}}{tf_j} \times \frac{1}{p_l}\right) \tag{7}$$

where MOFS(f_j, c_l) is the MOFS score of f_j with respect to c_l, w_l is the weight sum of the samples in category c_l, w_{jl} is the weight sum of the samples which contain f_j in category c_l, w_{jl}/w_l is the weighted intra-category score which denotes the importance of f_j with respect to c_l, p_l is a factor which is used to reduce the effect of category sizes. In order to normalize MOFS(f_j, c_l) to interval [0, 1], the factor p_l is defined as:

$$p_l = \frac{tc_l}{\min_{c_k \in C}(tc_k)} \tag{8}$$

where C is the set of all categories. Obviously, from formula (7) we know that a feature which has high category distinguishing ability and occurs in the samples with high weights tends to obtain a high score. Consequently, the details of MOFS are given as follows:

Algorithm 2: The executing process of MOFS.

Input: training sample matrix: $\mathbf{TS} \in \mathrm{R}^{M\times N}(\mathbf{TS} = [\mathbf{s}_i]$, $1 \le i \le M$, M and N are the numbers of samples and features in **TS**); label matrix of training samples: $\mathbf{Y} \in \mathrm{R}^{M\times 1}$; label set: $C=\{c_l\}$ $(1 \le l \le L)$; sample weight matrix of iteration t: $\mathbf{D}_t \in \mathrm{R}^{1\times M}$; minimum and maximum values in feature normalization: $x_{\min}$, $x_{\max}$; number of quantization levels in feature discretization: N_Q; number of selected features: n; the j_{th} feature: $f_j(1 \le j \le N)$.

Output: final selected features subset fs.

1: **for** i = 1: 1: M
2: **for** j = 1: 1: N
3: Normalize $\mathbf{s}_i$'s feature value of f_j (denoted as s_{ij}):$s_{ij} = x_{\min} + \frac{s_{ij} - \min(f_j)}{\max(f_j) - \min(f_j)} \times (x_{\max} - x_{\min})$, where max($f_j$) and min($f_j$) return the maximum and minimum values of f_j respectively.
4: Discrete $\mathbf{s}_i$ using the Equal Width Interval Binning (EWIB) algorithm [56].
5: **end for**
6: **end for**
7: **for** j = 1: 1: N
8: **for** k = 0:1:N_Q-1
9: Calculate the number that k occurs in f_j and denote it as tf_{jk}.
10: **end for**
11:**end for**
12:**for** l = 1: 1: L
13: Calculate the weight sum and the number of all samples in c_l, and denote them as w_l and n_l, respectively.
14: Calculate p_l using formula (8).
15: **for** j = 1: 1: N
16: **for** k = 0:1:N_Q-1

17: Calculate the weight sum and the number of the samples in which k occurs in f_j in c_l and denote them as w_{jlk} and tfc_{jlk}, respectively.
18: **end for**
19: **end for**
20:**end for**
21: **for** j = 1: 1: N
22: Set the MOFS score of f_j to $mofs_j$ = 0.
23: **for** k = 0:1:N_Q-1
24: $mofs_j = mofs_j + \sum_{c_l} \frac{w_{jlk}}{w_l} \times \frac{tfc_{jlk}}{tf_{jk}} \times \frac{1}{p_l}$
25: **end for**
26:**end for**
27:Rank all features in descend order according to their MOFS scores, and select the top n features to obtain the final selected feature subset fs.

4.2. The proposed AdaBoost algorithm

4.2.1. MOFS and WNMF based AdaBoost algorithm

Traditional AdaBoost algorithms like M1 [12], SAMME [14] and SAMMER [15] face the following problems: (1) Features selected by the traditional feature selection methods like IG and CMFSX are explicit. These methods cannot make full use of the part-based information of the original feature space, and often have to discard features which have important information. (2) According to step 7 of Algorithm 1 we know that only one classifier is used when updating the sample weights. All misclassified samples use the same scaling factor in each iteration. However, the misclassification degrees of these samples are often different and the misclassified samples with higher misclassification degrees should draw more attention than those with lower misclassification degrees in the next iteration.

WNMF has two advantages compared to the traditional feature selection methods: (1) it creates new features (part-based features) that contain information from all original features in an unsupervised way [51,52]; (2) it can handle the outliers and noises of the dataset [57]. For solving the first problem of traditional AdaBoost algorithms, we utilize the advantages of WNMF and combine it with the proposed MOFS method to both obtain the part-based features and the explicit features. When WNMF is applied, the basis patterns which are represented by the columns of matrix **W** are the part-based features, and the feature number is the dimension of the embedding space. In this case, each sample is represented as a linear combination of the extracted part-based features with coefficients given by the corresponding columns of matrix **H**. Given the entire training sample matrix $\mathbf{TS} \in \mathrm{R}^{M\times N}(\mathbf{TS} = [\mathbf{s}_i], 1 \le i \le M)$ with the label matrix $\mathbf{Y} \in \mathrm{R}^{M\times 1}$, in order to ensure that each element in **TS** is non-negative, we first map each element into the interval [0, 1] by using Min-Max Normalization, which is defined as follows:

$$x_{ij} = \frac{x_{ij} - x_{j,\min}}{x_{j,\max} - x_{j,\min}} \tag{10}$$

where x_{ij} is the j_{th} feature value of $\mathbf{s}_i$, $x_{j,\min}$ and $x_{j,\max}$ are the minimum and maximum of the j_{th} feature value. As the sampling method with replacement yields significantly better performance than the sampling method without replacement [21], we use it to select a subset of the normalized entire training sample matrix to form a non-negative training sample matrix $\mathbf{S}_t \in \mathrm{R}^{m\times N}$ ($\mathbf{S}_t = [\mathbf{s}_{ti}]$, $\mathbf{s}_{ti} \in \mathrm{R}^{1\times N}$ is the vector of i_{th} sample in $\mathbf{S}_t$, $1 \le i \le m$, m is the number of samples in $\mathbf{S}_t$ and is set to $m = M/2$ in this paper) by considering the sample weights of the t_{th} iteration. Moreover, we denote the label matrix of $\mathbf{S}_t$ as $\mathbf{Y}_t \in \mathrm{R}^{m\times 1}$. Then, according to formulas (4) and (5), we use WNMF to decompose the transposed matrix of $\mathbf{S}_t$ and obtain the non-negative matrices $\mathbf{W}_t \in \mathrm{R}^{N\times r}(\mathbf{W}_t = [\mathbf{w}_{tj}]$, $1 \le j \le N$, r satisfies $(N + m) \times r < N \times m))$ and $\mathbf{H}_t \in \mathrm{R}^{r\times m}(\mathbf{H}_t = [\mathbf{h}_{tl}]$, $1 \le l \le r)$, where the weighting matrix $\mathbf{P}_t \in \mathrm{R}^{m\times N}$ ($\mathbf{P}_t = [\mathbf{p}_{ti}]$, $1 \le i \le m$) is defined as follows:

$$\boldsymbol{P}_t = \begin{bmatrix} \boldsymbol{p}_{t1} \\ \boldsymbol{p}_{t2} \\ \vdots \\ \boldsymbol{p}_{tm} \end{bmatrix} = \begin{bmatrix} p_{t11} & p_{t12} & \cdots & p_{t1N} \\ p_{t21} & p_{t22} & \cdots & p_{t2N} \\ & \cdots & & \\ p_{tm1} & p_{tm2} & \cdots & p_{tmN} \end{bmatrix} = \begin{bmatrix} d(\boldsymbol{s}_{t1}) & d(\boldsymbol{s}_{t1}) & \cdots & d(\boldsymbol{s}_{t1}) \\ d(\boldsymbol{s}_{t2}) & d(\boldsymbol{s}_{t2}) & \cdots & d(\boldsymbol{s}_{t2}) \\ & \cdots & & \\ d(\boldsymbol{s}_{tm}) & d(\boldsymbol{s}_{tm}) & \cdots & d(\boldsymbol{s}_{tm}) \end{bmatrix} \tag{11}$$

where $d(\mathrm{x})$ denotes the sample weight of $\mathbf{x}$. Obviously, $\mathbf{W}_t$ is the basis matrix which consists of the basis vectors of r part-based features in $\mathbf{S}_t$, and $\mathbf{H}_t$ is the weight matrix corresponding to the part-based features in $\mathbf{W}_t$. Moreover, we know that $\mathbf{S}_t$'s column number is reduced from N to r and $\mathbf{H}_t^{\mathrm{T}}$ is the new representation of $\mathbf{S}_t$ by using the extracted r part-based features. We know from formula (4) that, compared to NMF, WNMF minimizes the loss on high-weight samples in matrix decomposition process by introducing the weighting matrix $\mathbf{P}_t$, thus can improve the representation accuracy and learning result of high-weight samples.

Consequently, we obtain the training sample matrix $\mathbf{S}_t^{\mathrm{f}} \in \mathrm{R}^{m\times n}$ using the features selected by MOFS. In order to avoiding the dataset imbalance problem, we use the STOME method [29] to balance the sizes of different categories in $\mathbf{S}_t^{\mathrm{f}}$ and $\mathbf{H}_t^T$, so as to obtain an explicit feature represented matrix $\mathbf{S}_{\mathrm{bt}} \in \mathrm{R}^{m1\times N}$($m_1$ is the number of samples in $\mathbf{S}_{\mathrm{bt}}$) with the label matrix $\mathbf{Y}_{t1} \in \mathrm{R}^{m1\times 1}$and a part-based feature represented matrix $\mathbf{H}_{\mathrm{bt}} \in \mathrm{R}^{m_2\times N}$($m_2$ is the number of samples in $\mathbf{H}_{\mathrm{bt}}$) with the label matrix $\mathbf{Y}_{t2} \in \mathrm{R}^{m_2\times 1}$. On this basis, we obtain two trained models $h_{\mathrm{a}t}$ and $h_{\mathrm{b}t}$ by using two typical classifiers h_{a} and h_{b}($h_{\mathrm{a}} \neq h_{\mathrm{b}}$):

$$h_{\mathrm{a}t} = \mathrm{TR}(h_{\mathrm{a}}, \boldsymbol{S}_{\mathrm{bt}}, \boldsymbol{Y}_{t1}), h_{\mathrm{b}t} = \mathrm{TR}(h_{\mathrm{b}}, \boldsymbol{H}_{\mathrm{bt}}, \boldsymbol{Y}_{t2}) \tag{12}$$

According to [58] we know that, for any sample $\mathbf{s}_i \in \mathrm{R}^{N\times 1}$ $(1 \le i \le M)$ in **TS**, we can obtain a feature vector $\mathbf{x}_i^{\mathrm{NMF}} \in \mathrm{R}^{r\times 1}$ which is the projection result of $\mathbf{s}_i$ using the part-based features contained in $\mathbf{W}_t$. Obviously, $\mathbf{x}_i^{\mathrm{NMF}}$ satisfies:

$$\begin{aligned} &\boldsymbol{s}_i \approx \boldsymbol{W}_t \boldsymbol{x}_i^{\mathrm{NMF}} \\ &s.t. \mathbf{W}_{\mathrm{t}} \geqslant 0, \mathbf{x}_{\mathrm{i}}^{\mathrm{NMF}} \geqslant 0 \end{aligned} \tag{13}$$

As $\mathbf{W}_t$ may not have inverse matrix, we obtain the pseudo-inverse of $\mathbf{W}_t$ which is denoted as $\mathbf{W}_t^{+}(\mathbf{W}_t^{+} \in \mathrm{R}^{r\times N})$ and apply it to project $\mathbf{s}_i$ into $\mathbf{x}_i^{\mathrm{NMF}}$. The corresponding formula is shown as follows:

$$\begin{cases} \boldsymbol{W}_t^{+} = \left(\boldsymbol{W}_t^{\mathrm{T}} \boldsymbol{W}_t\right)^{-1} \boldsymbol{W}_t^{\mathrm{T}} \\ \boldsymbol{x}_i^{\mathrm{NMF}} = \boldsymbol{W}_t^{+} \boldsymbol{s}_i \end{cases} \tag{14}$$

Suppose $\boldsymbol{x}_i^{\mathrm{FS}} \in \mathrm{R}^{n\times 1}$ is the representation of $\mathbf{s}_i$ using n selected features obtained by MOFS, the predicted labels of $\mathbf{s}_i$ using the trained classifiers $h_{\mathrm{a}t}$ and $h_{\mathrm{b}t}$ in the t_{th} iteration are:

$$\begin{cases} l_{\mathrm{a}ti} = h_{\mathrm{a}t}\left(\boldsymbol{x}_i^{\mathrm{FS}}\right) \\ l_{\mathrm{b}ti} = h_{\mathrm{b}t}\left(\boldsymbol{x}_i^{\mathrm{NMF}}\right) \end{cases} \tag{15}$$

Further, in order to solve the problem that traditional sample weight updating method treats all misclassified samples equally, we introduce the concept of misclassification degree and differentiate the misclassified samples by evaluating their misclassification degrees. The purpose of the improvement is to differentiate the scaling factors of misclassified samples in each iteration, so as to assign different weights to samples according to the classification results of h_{a} and h_{b}. For any$\boldsymbol{s}_i \in \boldsymbol{TS}$, we define its misclassification degree δ_i as follows:

通过MOFS得分，选择前n个特征作为显示特征

$$\begin{cases} \delta_i = \alpha, \text{if} h_{at}(\mathbf{x}_i^{FS}) == h_{bt}(\mathbf{x}_i^{NMF}) \neq \mathbf{Y}(\mathbf{s}_i) \\ \delta_i = \beta, \text{else if} h_{at}(\mathbf{x}_i^{FS}) \neq h_{bt}(\mathbf{x}_i^{NMF}) \\ \delta_i = \chi, \text{else} \end{cases} \tag{16}$$

where α, β and χ are parameters which adjust the scaling factors of the misclassified samples. In this paper, α, β and χ are parameters and set to be α = 1, β = 0.5 and χ = 0 respectively. On this basis, as is shown in formula (17), we propose a fine grained sample weight updating method which considers the misclassification degree information based on the traditional sample weight updating method.

$$\begin{cases} d_{t+1,i} = d_{t,i} e^{\delta_i \alpha_t} / Z_t \\ Z_t = \sum_{i=1}^{M} d_{t,i} e^{\delta_i \alpha_t} \\ \alpha_t = (\alpha_{at} + \alpha_{bt})/2 \end{cases} \tag{17}$$

Where $d_{t,i}$ is the sample weight of $\mathbf{s}_i$, δ_i is the misclassification degree of $\mathbf{s}_i$, Z_t is the normalization factor, α_{at} and α_{bt} of which the calculation method is shown in section 4.2.2 are the classifier weights of h_{at} and h_{bt}.

Clearly, the traditional sample weight updating method shown in step 10 of Algorithm 1 is just a special case of the proposed method without considering the situation that δ_i equals β. According to formula (16) we know that, suppose δ_i and δ_j are the misclassification degrees of two samples $\mathbf{s}_i$ and $\mathbf{s}_j$ in iteration t, we have $d_{t+1,i} > d_{t+1,j}$ if there exists $\delta_i > \delta_j$. Therefore, we deduce that the proposed fine grained sample weight updating method can distinguish different misclassified samples and give more attentions to the misclassified samples with high misclassification degrees.

4.2.2. Calculation of the classifier weights α_{at} and α_{bt}

From Algorithm 1 we know that the traditional classifier weight calculation method will increase the weights of the samples that are misclassified in each iteration. These samples may be misclassified many times, which causes the weights of the samples to increase continuously and results in a serious imbalance of the sample weight distribution [19]. This deduces the problem that the weight of the current classifier cannot be calculated properly as the traditional classifier weight calculation method only considers the samples weights of misclassified samples. We now take a simple example to show the problem. Suppose there exits eight samples s_1-s_8, as is shown in Fig. 1, a square or a circle denotes a sample of which the real label is c_1 or c_2. Given the weights of these samples in the t_{th} iteration, we apply-two different classifiers h_a and h_b to s_1-s_8 to obtain the trained classifiers h_{at} and h_{bt}, and the corresponding classification results of these samples are shown in Fig. 1(a) and Fig. 1(b) respectively (the classification boundaries are denoted in dashed straight lines). In Fig. 1, the samples with color blue or pink denote that they are classified into c_1 or c_2. Obviously, as the performance of a classifier relies on the training model and the corresponding parameters, it is very probable that the results of h_{at} and h_{bt} are quite different in practical applications.

From Fig. 1 we know that, when a traditional AdaBoost algorithm like AdaBoost.M1 or SAMME is used, the corresponding classifier error rates are ε_{at} = ε_{bt} = 0.30. According to steps 7 and 8 of Algorithm 1 we have α_{at} = α_{bt} = ln((1–0.30)/0.30)≈0.847, which means that h_a and h_b have the same classification ability in the current iteration. However, it is obvious that most of the samples in c_2 are misclassified when h_b is used and only one sample in c_2 is misclassified when h_a is used. Thus, the results obtained using the traditional classifier weight calculation method is contrary to the fact that the classification ability of h_a is higher than that of h_b as h_a can correctly classify more samples. Through the example, we can draw the conclusion that the traditional classifier weight calculation method deduces the problem that a poorly performing classifier may be assigned a high weight as this method only considers the weights of misclassified samples when calculating the classifier error rate.

For solving this problem, we consider the classification performance of the current classifier on the entire training set, and propose a new classifier weight calculation method which utilizes the advantage of macro F_1 measurement [21] on evaluating the classification ability of a classifier with respect to each category. According to [21], the macro F_1 measurement is defined as follows:

$$\begin{cases} F_1 = \frac{1}{L}\sum_{i=1}^{L} \frac{2 \times re_i \times pr_i}{re_i + pr_i} \\ re_i = \frac{TP_i}{TP_i + FN_i} \\ pr_i = \frac{TP_i}{TP_i + FP_i} \end{cases} \tag{18}$$

where re_i and pr_i are the recall and precision of a classifier on category c_i respectively, L denotes the number of categories, TP_i is the number of samples which are correctly classified into category c_i, FP_i is the number of samples which are misclassified into category c_i, FN_i is the number of samples which belong to category c_i and are misclassified. In order to both consider a classifier's performances on misclassified samples and all samples, we combine the traditional classifier weight calculation method and the macro F_1 measurement to calculate the classifier weight which belongs to interval [0, 1) in each iteration. According to steps 7 and 8 of Algorithm 1, we first calculate the classification error rates ε_{at} and ε_{bt} of trained classifiers h_{at} and h_{bt}, so as to obtain the corresponding classifier weights α_{oat} and α_{obt}. As there exists $\varepsilon_{at} < (L-1)/L$, we have:

$$\begin{aligned} \alpha_{oat} &= \ln\left(\frac{1-\varepsilon_{at}}{\varepsilon_{at}}\right) + \ln(L-1) > \ln\left(\frac{1}{L-1}\right) + \ln(L-1) \\ &= \ln(1) = 0 \end{aligned} \tag{19}$$

Obviously, α_{oat} and α_{obt} satisfy $\alpha_{oat} > 0$ and $\alpha_{obt} > 0$. Furthermore, in order to better balance the traditional classifier weight calculation method and the macro F_1 measurement, as is shown in formula (20), we use the Sigmoid function to map α_{oat} and α_{obt} to Γ_{at} and Γ_{bt} respectively.

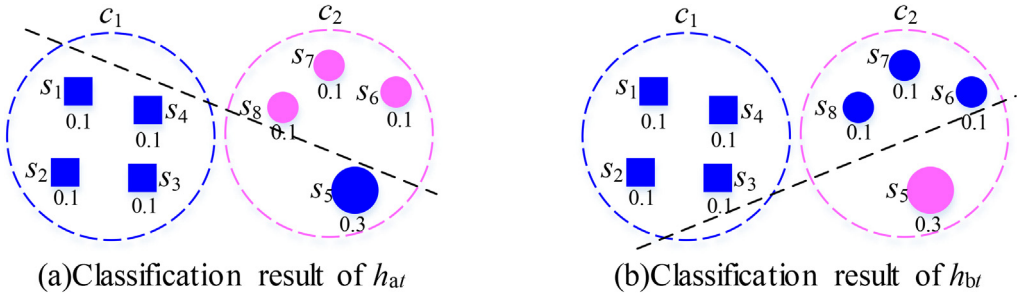

Fig. 1. Results of h_{at} and h_{bt} when traditional classifier weight calculation method is used.

$$\begin{cases} \Gamma_{at} = \frac{2}{1+e^{-\alpha_{oat}}} - 1 = \frac{1-e^{-\alpha_{oat}}}{1+e^{-\alpha_{oat}}} \\ \Gamma_{bt} = \frac{2}{1+e^{-\alpha_{obt}}} - 1 = \frac{1-e^{-\alpha_{obt}}}{1+e^{-\alpha_{obt}}} \end{cases} \tag{20}$$

Obviously, there exists $\Gamma_{at} \in (0, 1)$ and $\Gamma_{bt} \in (0, 1)$. Then, we calculate F_{at} and F_{bt} which are the macro F_1 values of h_{at} and h_{bt} on the entire dataset respectively. On this basis, we obtain the classifier weights of h_{at} and h_{bt} (denoted as α_{at} and α_{bt}) as follows:

$$\alpha_{at} = \Gamma_{at} F_{at}, \alpha_{bt} = \Gamma_{bt} F_{bt} \tag{21}$$

Obviously, there exists $F_{at} \in [0, 1]$ and $F_{bt} \in [0, 1]$, thus we have $\alpha_{at} \in [0,1)$ and $\alpha_{bt} \in [0,1)$. It can be easily seen from formula (21) that the proposed classifier weight calculation method not only considers the importance of the classifiers which can distinguish the samples with high weights, but also considers the importance of the classifiers which have good performances on distinguishing all samples. To verify the effectiveness of this method, we apply it to the samples of Fig. 1 and obtain: $\alpha_{oat} = \alpha_{obt} \approx 0.847$, $\Gamma_{at} = \Gamma_{bt} \approx \frac{1-e^{-0.847}}{1+e^{-0.847}} \approx 0.399$, $F_{at} \approx 0.873$, $F_{bt} \approx 0.564$. Thus, we have $\alpha_{at} = 0.399 \times 0.873 \approx 0.348 > \alpha_{bt} = 0.399 \times 0.564 \approx 0.225$. Obviously, the result of the proposed classifier weight calculation method is consist with the fact that the classification performance of h_a is better than that of h_b, showing that our method is much more reasonable than the traditional classifier weight calculation method when measuring the classifier importance.

4.2.3. Main steps of the proposed AdaBoost algorithm

As is shown in Fig. 2, the pseudo-code of the proposed AdaBoost algorithm is summarized in Algorithm 3. It can be seen from Algorithm 3 that the advantages of the proposed AdaBoost algorithm over the traditional AdaBoost algorithms mainly lie in three aspects: (1) The proposed algorithm proposes the MOFS method and selects the most discriminative explicit features by considering the sample weight information. (2) Through the combination of MOFS and WNMF, not only the explicit features but also the part-based features are considered in each iteration of AdaBoost. (3) for the purpose of distinguishing different misclassified samples, the concept of misclassification degree is introduced and a fine grained sample weight updating method is proposed. Moreover, the macro F_1 measurement is combined in the classifier weight calculation process to solve the problem that a classifier with lower classification ability may be greatly highlighted in subsequent iterations by traditional methods.

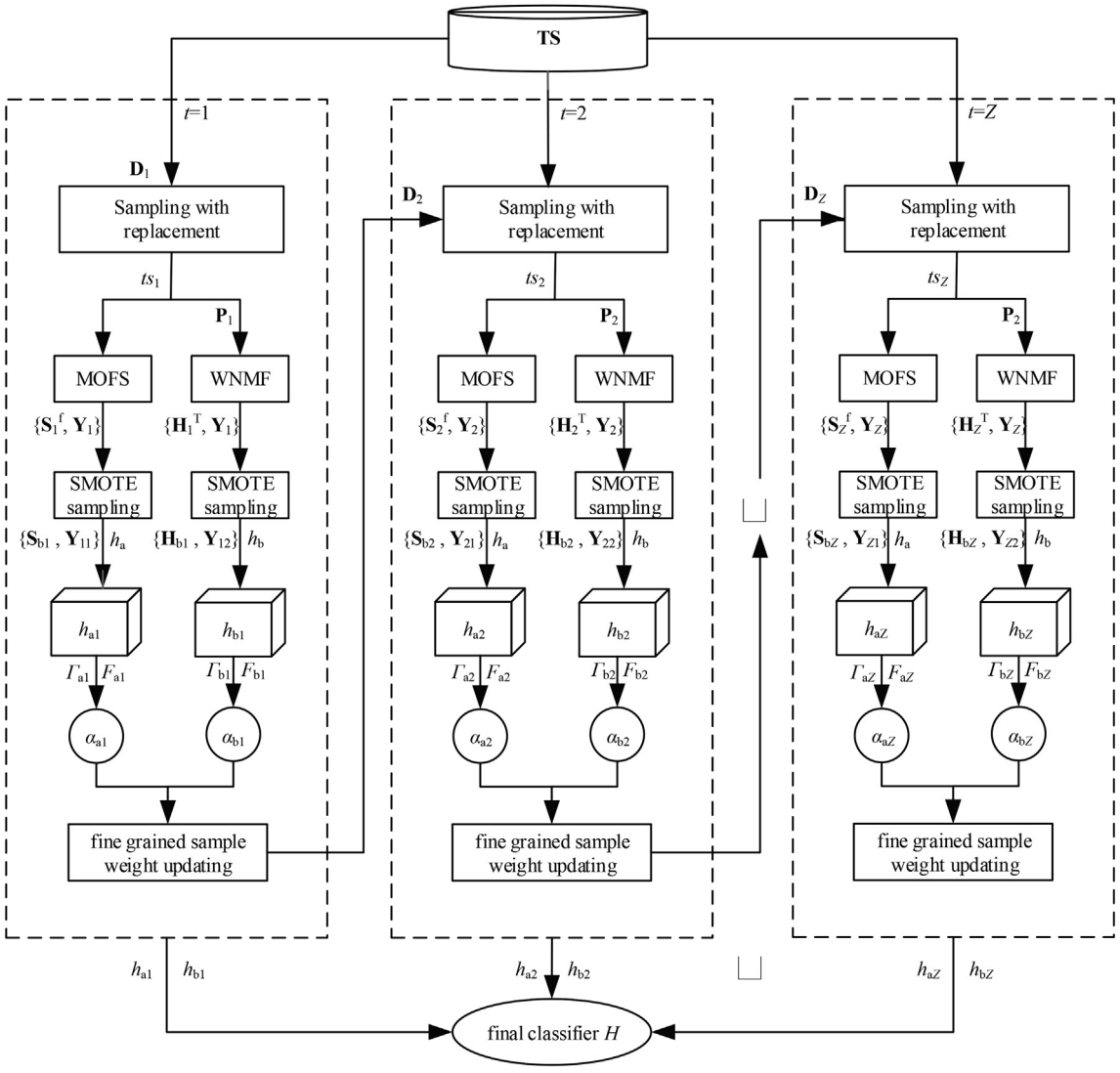

Fig. 2. Flow chart of the proposed AdaBoost algorithm.

Algorithm 3: Proposed AdaBoost algorithm based on MOFS and WNMF.

Input: training sample matrix: $\mathbf{TS} \in \mathrm{R}^{M\times N}(1 \le i \le M$, M and N are the numbers of samples and features in **TS**); label matrix of training samples: $\mathbf{Y} \in \mathrm{R}^{M\times 1}$; testing sample matrix: $\mathbf{TE} \in \mathrm{R}^{U\times N}(U$ is the number of testing samples); label set of training samples: $C=\{c_i\}(1 \le i \le L$, L is the number of categories); number of features selected by MOFS: n; number of iterations: Z; sample weight matrix: $\mathbf{D} = [\mathbf{D}_t] \in \mathrm{R}^{Z\times M}(1 \le t \le Z)$; weak classifiers: h_a, h_b; classifier training function: **TR**.

Output: the final classifier H.

1: Initialize a flag array **is_valid**: **is_valid**$[Z]$=[1].
2: **for** each value $d_{1,i}$ **in** $\mathbf{D}_1$
$d_{1,i} = 1/M$.
3: **end for**
4: **for** $t = 1{:}\ 1{:}\ Z$
5: Select $m(m = M/2)$ samples from **TS** using the sampling method with replacement [21] to form a sub sample matrix $\mathbf{S}_t$ according to the sample weights of $\mathbf{D}_t$.
Obtain $\mathbf{S}_t$'s label vector $\mathbf{Y}_t$ and construct the training sample subset ts_t=$\{\mathbf{S}_t, \mathbf{Y}_t\}$.
Select the feature subset fs_t from ts_t using Algorithm 2, and represent $\mathbf{S}_t$ as $\mathbf{S}_t^{\mathrm{f}}$ using fs_t.
Apply the SMOTE sampling method [29] to ts_t=$\{\mathbf{S}_t^{\mathrm{f}}, \mathbf{Y}_t\}$ to obtain a balanced training set ts'_t=$\{\mathbf{S}_{\mathrm{bt}}, \mathbf{Y}_{t1}\}$.
Apply h_a to ts'_t to obtain the trained classifier h_{at} according to formula (12).
Obtain $\mathbf{P}_t$ and apply WNMF to $\mathbf{S}_t^{\mathrm{T}}$ to obtain $\mathbf{W}_t$ and $\mathbf{H}_t$ according to formula (5).
Apply the SMOTE method to ts_t=$\{\mathbf{H}_t^{\mathrm{T}}, \mathbf{Y}_t\}$ to obtain a balanced training set ts'_t=$\{\mathbf{H}_{\mathrm{bt}}, \mathbf{Y}_{t2}\}$.
Apply h_b to ts'_t to obtain the trained classifier h_{bt} according to formula (12).
for each sample s_i **in TS**
Use formula (15) to get s_i's predicted labels l_{ati} and l_{bti}.
end for
Calculate ε_{at} and ε_{bt}.
if $\varepsilon_{\mathrm{at}} \ge (L\text{-}1)/L$ or $\varepsilon_{\mathrm{bt}} \ge (L\text{-}1)/L$
is_valid$[t] = 0$, continue.
end if
Calculate Γ_{at}, Γ_{bt}, F_{at}, F_{bt}, α_{at} and α_{bt} according to formulas (19)-(21).
for each sample s_i **in** $\mathbf{S}_t$
Calculate δ_i and apply the fine grained sample weight updating method to update the weight of s_i according to formulas (16) and (17).
end for
end for
for each sample s_i **in TE**
for $t = 1{:}\ 1{:}\ Z$
Represent s_i using fs_t to obtain$\boldsymbol{x}_{ti}^{\mathrm{FS}}$, and project s_i to $\boldsymbol{x}_{ti}^{\mathrm{NMF}}$ using the basis matrix $\mathbf{W}_t$ according to formula (13).
end for
Obtain s_i's final label $H(s_i)$:

$$H(s_i) = \arg\max_{c_j \in C} \begin{pmatrix} \sum_{t=1}^{Z} \alpha_{\mathrm{at}} \mathrm{I}(\boldsymbol{is\_valid}[t] == 1 \&\& h_{at}(\boldsymbol{x}_{ti}^{\mathrm{FS}}) == c_j) \\ + \sum_{t=1}^{Z} \alpha_{\mathrm{bt}} \mathrm{I}(\boldsymbol{is\_valid}[t] == 1 \&\& h_{bt}(\boldsymbol{x}_{ti}^{\mathrm{NMF}}) == c_j) \end{pmatrix}$$

end for

5. Experimental results and analysis

5.1. Experimental setup

Experiments are performed based on eight typical datasets which are obtained from the UCI Machine Learning Repository [59,60]. With respect to each dataset, we give the sample number, feature number, category number, maximum sample number of all categories and minimum sample number of all categories in Table 1.

We implement all algorithms on the platform of 64G memory and Intel Xeon Gold 5218 CPU by using Python 3.6. Similar to other studies, we randomly select 90 % and 10 % of the samples in each dataset as the training set and the testing set respectively. The iteration number Z is set to $Z = 10$ in case of no special statement, and the parameters of WNMF are given as follows: Loss function: Frobenius norm, maximum number of iterations: $ZW = 200$. In steps 3 and 4 of Algorithm 2, $x$min and $x$max are set to 0 and 6 respectively, and NQ is set to $NQ = x\mathrm{max}$. Moreover, as the macro F1 measurement [21] synthetically considers the precision and recall, we use it to compare the classification abilities of different AdaBoost based algorithms.

5.2. Selection of value r in WNMF

According to [54] we know that, the value r in WNMF represents the number of part-based features and is usually decided by experiments and experience. In order to achieve the best r value with respect to different datasets, we introduce the ratio factor $\lambda \in [0, 1]$ and set r to $r = \max\left\{\mathrm{floor}\left(\frac{\lambda NM}{N+M} - 1\right), 1\right\}$. On this basis, we test the performances of three typical classifiers (SVM, KNN and RF) when the part-based features are considered in Algorithm 1(step 6 of Algorithm 1 is modified and the trained classifier ht is obtained according to steps 10–12 in Algorithm 3). For ease of comparisons, λ is set to range from 0.1 to 1.0 with a step of 0.1, and the parameters of these classifiers are given as follows: (1) SVM: kernel function: RBFKernel, penalty coefficient: $C = 400$, kernel function parameter $g = 0.07$; (2) RF: number of trees: $n$tr = 5; (3) KNN: number of neighbours: $k = 5$. On this basis, the average macro F1 values (Fa) and average executing time values (eta) are calculated when experiments are carried out with respect to each dataset for 100 times, and the results are shown in Fig. 3. From Fig. 3 we know that the eta values are generally increasing as λ ranges from 0.0 to 1.0, showing that the running speeds of the above classifiers are greatly affected by r. Moreover, it is obvious that the Fa values are fluctuating with different degrees as the λ value increases, illustrating that r may deduce low classification accuracy if it is too large or too small. Further, we notice that all classifiers can achieve relatively high Fa values under the premise of keeping low eta values when $\lambda = 0.2$. In order to balance the classification accuracy and the executing speed, we set λ to $\lambda = 0.2$ and r to $r = \max\left\{\mathrm{floor}\left(0.2 \times \frac{NM}{N+M} - 1\right), 1\right\}$ in this paper.

5.2.1. Selections of classifiers h_a and h_b

In order to determine the classifiers ha and hb, three widely used classifiers: RF, SVM and KNN are tested on eight datasets. When h_a and h_b are instantiated to RF, SVM and KNN respectively,

Table 1
Information of experimental datasets.

Dataset	Sample number	Feature number	Category number	Maximum sample number of all categories	Minimum sample number of all categories
Sonar	208	60	2	111	97
Amazon	300	10,000	10	30	30
Dermatology	366	34	6	112	20
Arrhythmia	452	279	13	245	2
DrivFace	606	6400	3	546	27
Splice	979	60	2	517	462
CNAE9	1080	857	9	120	114
Gisette	6000	5000	2	3000	3000

the average macro F1 values (called *F*a) of the proposed algorithm are obtained as the ratio of selected features ranges from 0.1 to 0.5 with a step of 0.1, and the results are shown in Fig. 4(a)-Fig. 4(h). For ease of comparison, the parameters of these classifiers are the same as those given in section 5.2. We first determine the best selection of *h*b by setting *h*a to RF, SVM and KNN respectively. From Fig. 4(a) to Fig. 4(h) we know that, when *h*a is set to RF, the combination of RF + SVM achieves the highest *F*a value for 6 times, which is obviously higher than those of RF + RF and RF + KNN. When *h*a is set to SVM, the methods of SVM + RF, SVM + SVM and SVM + KNN achieve the highest *F*a values for 2, 4 and 2 times respectively. When *h*a is instantiated to KNN, KNN + SVM performs significantly better than those of the other methods as it achieves the highest *F*a value for 4 times. Therefore, we conclude that SVM achieves high performance on classifying the samples represented by part-based features, and it is the best choice to set *h*b to SVM in the proposed algorithm.

Further, we determine the best *h*a when *h*b is set to SVM. It can be easily seen from Fig. 4 that, RF + SVM achieves the highest *F*a value for 5 times, while SVM + SVM and KNN + SVM achieve the highest *F*a value for twice and once respectively, deducing that RF is the best choice for dealing with samples represented by explicit features. Therefore, *h*a and *h*b are set to RF and SVM respectively in the proposed algorithm.

5.3. Comparisons of MOFS with typical feature selection methods

In order to validate the effectiveness of the proposed feature selection method, we compare MOFS with five typical feature selection methods (IG [38], CMFSX [39], CHI [41], TSFS [61] and GOPSO [62]) as the ratio of selected features (*rf*) ranges from 0.1 to 0.5 with a step of 0.1. The above feature selection methods are used in step 7 of Algorithm 3 respectively, and CMFSX is chose as the optimal feature selection of TSFS. When Algorithm 3 is executed for 100 times with respect to each feature selection method, the average macro F1 values (denoted as *F*a) are calculated and shown in Fig. 5. It can be seen from Fig. 5 that the performance of TSFS is generally better than CMFSX on the datasets of Sonar, Dermatology, DrivFace and CNAE9, and MOFS outputs the other methods in most cases. For example, MOFS outperforms CMFSX significantly except when the datasets of Splice and Gisette are used, and it achieves the highest improvement of about 0.15 when Sonar dataset is used. MOFS obtains higher *F*a values than those of the other methods in all cases when Amazon dataset is used, and it obtains the best performance except for the case of *rf* = 0.2 when the datasets of Arrhythmia and CNAE9 are used. Therefore, as MOFS considers the sample weight information which is helpful in selecting the features those can distinguish the misclassified samples in each iteration, it can improve the classification accuracy of AdaBoost significantly when comparing with the typical feature selections like IG, CMFSX, CHI, TSFS and GOPSO.

5.4. Comparisons of the different classifier weight calculation methods and sample weight updating methods

In order to validate the performances of the proposed classifier weight calculation method and fine grained sample weight updating method, we compare the following AdaBoost algorithms which are modified from Algorithm 3 by combining different classifier weight calculation methods and different sample weight updating methods:

(1) ADA: ADA derives from Algorithm 3 and uses the classifier weight calculation method of SAMME to get α_{at} and α_{bt}. Further, as is shown in formula (22), ADA uses the classifier weight α_t=(α_{at} + α_{bt})/2 to update the sample weights by utilizing the sample weight updating method of SAMME.

$$\begin{cases} d_{t+1,i} = \frac{d_{t,i}e^{\alpha_t \mathrm{I}(h_{at}(s_i)\neq \boldsymbol{Y}(s_i)||h_{bt}(s_i)\neq \boldsymbol{Y}(s_i))}}{Z_t} \\ Z_t = \sum_{i=1}^{M} d_{t,i}e^{\alpha_t \mathrm{I}(h_{at}(s_i)\neq \boldsymbol{Y}(s_i)||h_{bt}(s_i)\neq \boldsymbol{Y}(s_i))} \end{cases} \tag{22}$$

Where || represents the logical or operation.

(2) FG_ADA: FG_ADA derives from Algorithm 3 and applies the classifier weight calculation method of SAMME to get α_{at} and α_{bt}. Further, as is shown in formulas (16) and (17), FG_ADA updates the sample weights by utilizing the proposed fine grained sample weight updating method.

(3) FM_ADA: FM_ADA derives from Algorithm 3 and applies the proposed macro F_1 measurement based classifier weight calculation method to get α_{at} and α_{bt}. Further, FM_ADA uses the classifier weight α_t=(α_{at} + α_{bt})/2 to update the sample weights by utilizing formula (22).

(4) FG_FM_ADA: the proposed algorithm shown in Algorithm 3.

When the ratio of selected features obtained by MOFS ranges from 0.1 to 0.5 with a step of 0.1, each of the above algorithms is executed for 100 times, and the average macro F_1 values (denoted as F_a) are calculated and shown in Fig. 6. From Fig. 6 we observe that ADA performs the worst except when the datasets of DrivFace and CNAE9 are used. When comparing FG_FM_ADA with FG_ ADA, we find that the F_a values of the former algorithm are higher than those of the later one except when Dermatology is used, illustrating that it is very helpful to improve the classification ability by combining the macro F_1 measurement when calculating the classifier weights. Moreover, it is obvious that FG_FM_ADA performs better than FM_ADA on 6 datasets expect when Arrhythmia and CNAE9 are used, verifying the effectiveness of the proposed fine grained sample weight updating method on dealing with the misclassified samples compared to the traditional sample weight updating method.

5.5. Comparisons of the proposed algorithm with typical AdaBoost based algorithms

In this section, we combine six typical AdaBoost based algorithms (M1 [12], SAMME [14], SAMMER [15], M_ADA_A [20],

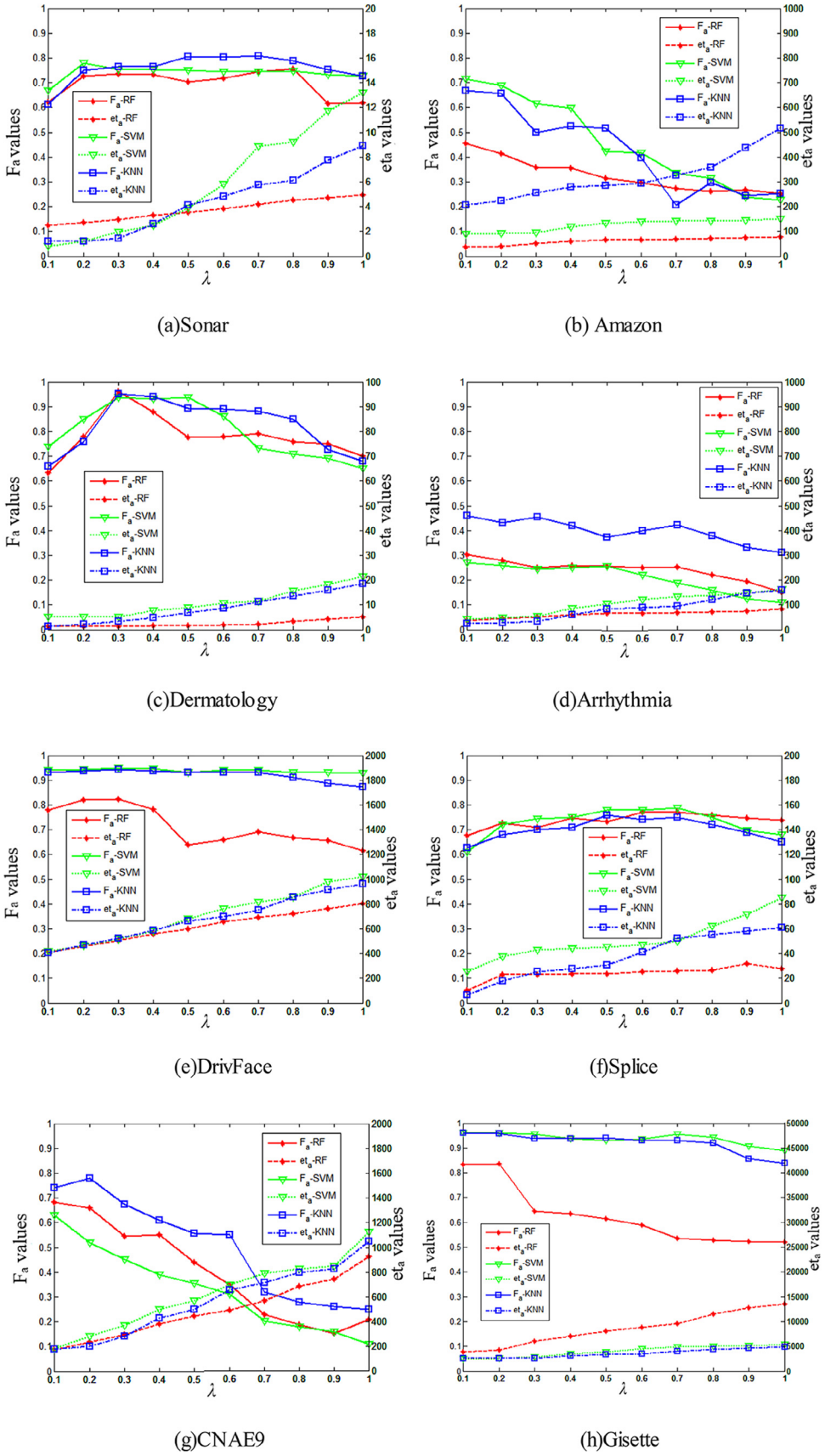

Fig. 3. Performances of three typical classifiers when only the part-based features are considered.

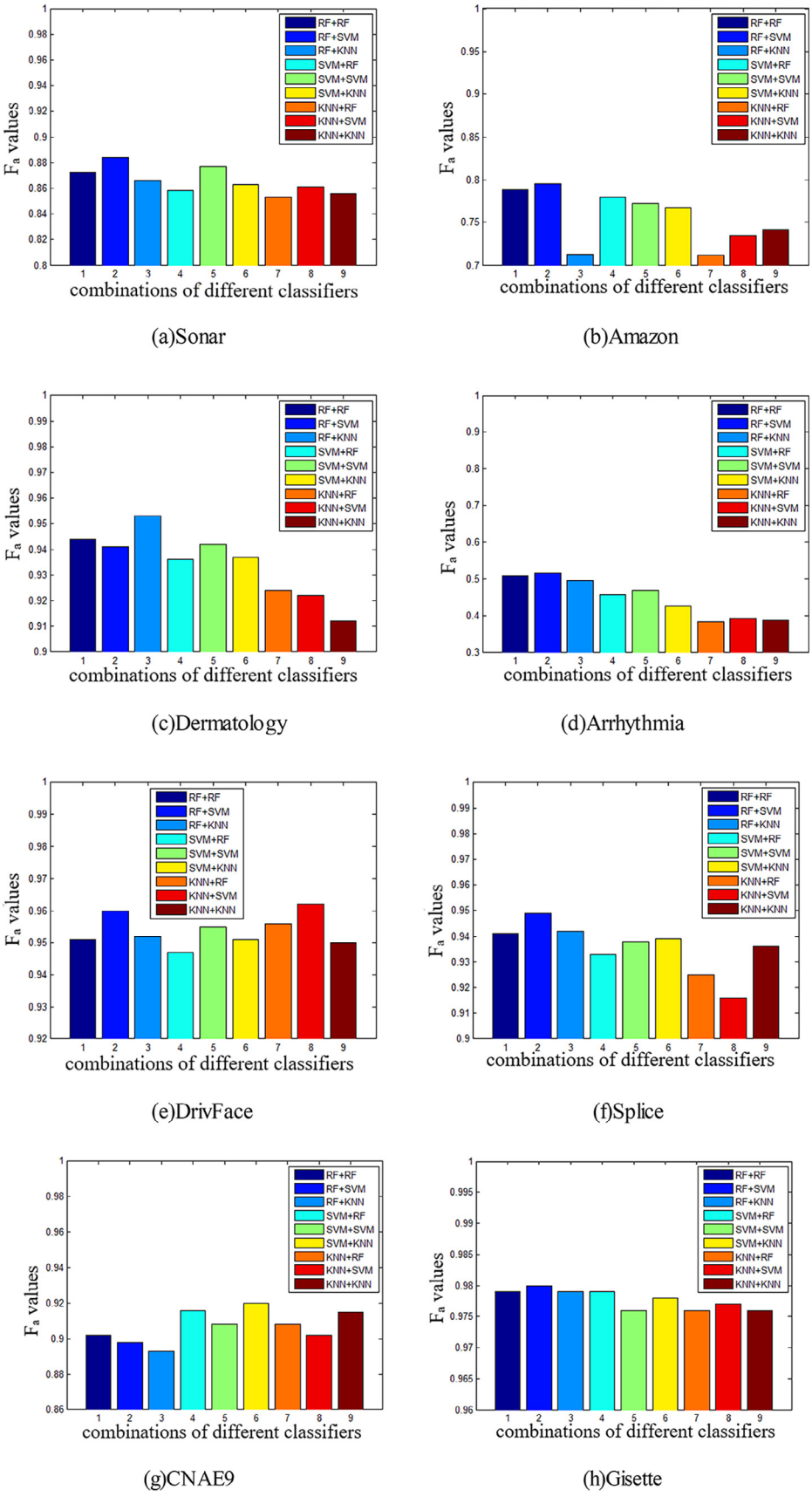

Fig. 4. *Fa* values when different classifiers are used on each dataset.

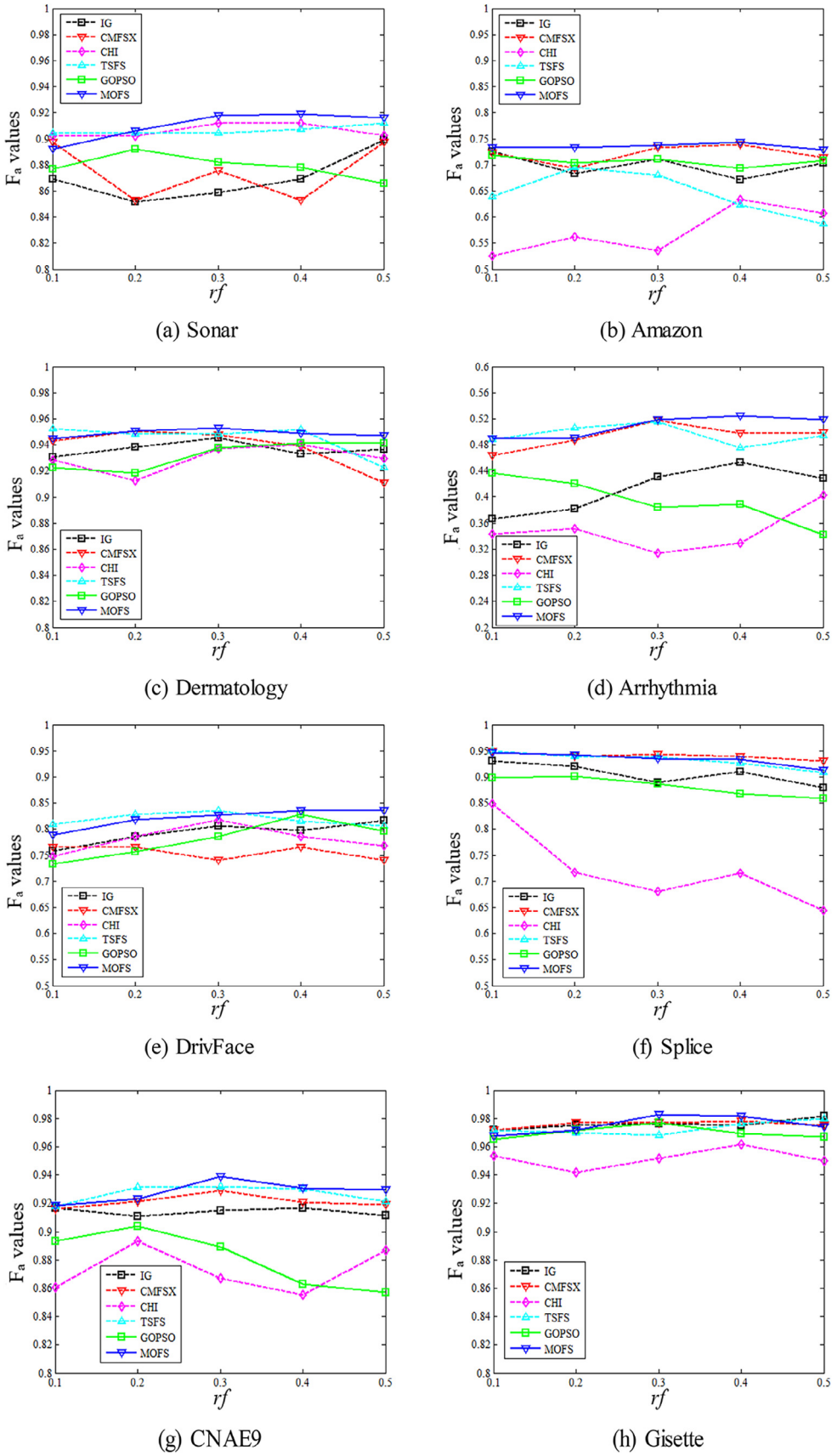

Fig. 5. Comparison of different feature selection methods.

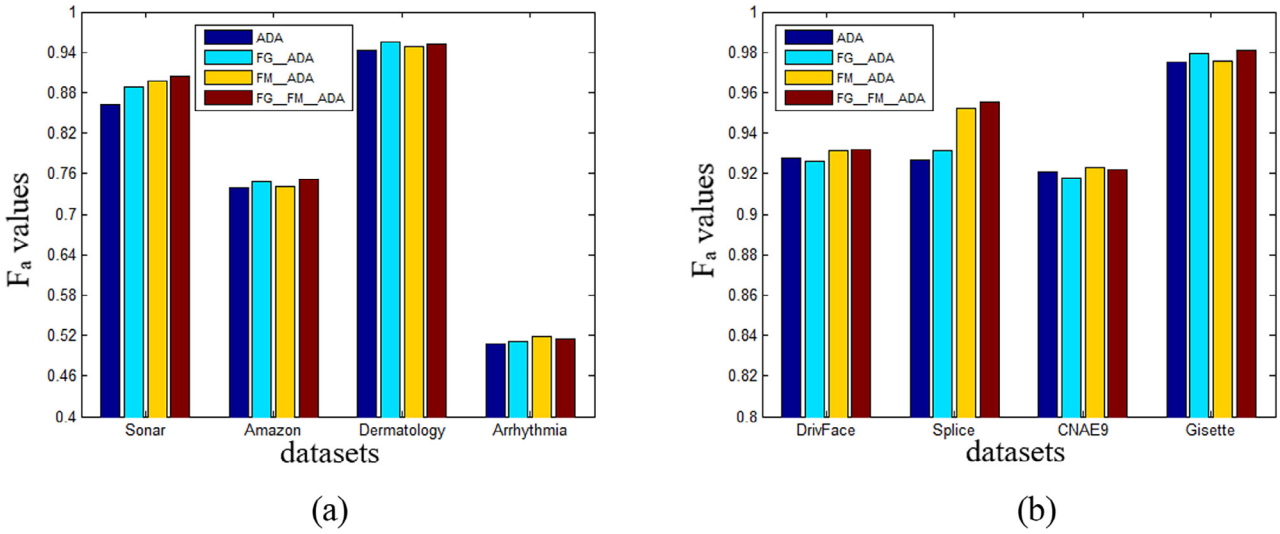

Fig. 6. Comparisons of different sample weight updating methods using different classifier weight calculation methods.

M_ADA_A_SMV [21] and MF_ADA [19]) with RF and SVM respectively, and compare the performances of these combined algorithms with that of the proposed algorithm. For the sake of fairness, some parameters are initialized first: in M_ADA_A_SMV, the number of expert classifiers is set to 10; in the above six typical AdaBoost based algorithms, the CMFSX method is applied to select the best features, and the parameters of RF and SVM are set according to section 5.2.

On this basis, as is shown in Tables 2-9, we obtain the average macro F1 value (denoted as *F*a) of each algorithm as the ratio of selected features ranges from 0.1 to 0.5 with a step of 0.1 when the iteration number *Z* ranges from 10 to 50 with a step of 10, respectively. Further, we calculate the average *F*a values (denoted as *F*aa) of different *Z* values for each algorithm and also give the results in Tables 2-9. For ease of comparison, the highest *F*a values with respect to each *Z* value and the top three highest *F*aa value for

Table 2
Comparison of different algorithms on dataset Sonar.

Algorithms	F_a					F_{aa}	et_a
	Z = 10	Z = 20	Z = 30	Z = 40	Z = 50		
M1 + RF	0.781	0.822	0.835	0.842	0.853	0.827(9)	2.328
M1 + SVM	0.852	0.868	0.875	0.858	0.871	0.865(5)	2.295
SAMME + RF	0.783	0.812	0.823	0.831	0.860	0.822(12)	2.31
SAMME + SVM	0.859	0.862	0.868	0.871	0.879	0.868(4)	2.226
SAMMER + RF	0.781	0.806	0.841	0.852	0.854	0.827(10)	2.328
SAMMER + SVM	0.836	0.865	0.875	0.883	0.886	**0.869(3)**	2.31
M_ADA_A + RF	0.771	0.796	0.826	0.851	0.879	0.825(11)	2.304
M_ADA_A + SVM	0.756	0.802	0.821	0.836	0.852	0.813(13)	2.274
M_ADA_A_SMV + RF	0.793	0.823	0.869	0.866	0.871	0.844(7)	**47.016**
M_ADA_A_SMV + SVM	**0.876**	0.868	0.873	0.892	**0.899**	**0.882(2)**	**45.402**
MF_ADA + RF	0.849	0.838	0.821	0.836	0.844	0.838(8)	2.244
MF_ADA + SVM	0.800	0.851	0.876	0.878	0.880	0.857(6)	2.223
Proposed algorithm	0.852	**0.902**	**0.905**	**0.896**	0.887	**0.888(1)**	2.587

Table 3
Comparison of different algorithms on dataset Amazon.

Algorithms	F_a					F_{aa}	et_a
	Z = 10	Z = 20	Z = 30	Z = 40	Z = 50		
M1 + RF	0.595	0.628	0.632	0.648	0.643	0.629(13)	1389.984
M1 + SVM	0.702	0.705	0.718	0.721	0.738	0.717(9)	**1742.157**
SAMME + RF	0.638	0.665	0.679	0.708	0.716	0.681(11)	1425.618
SAMME + SVM	0.714	0.631	0.766	0.775	0.786	0.734(6)	**2273.439**
SAMMER + RF	0.645	0.685	0.678	0.719	0.708	0.687(10)	**1922.61**
SAMMER + SVM	0.736	0.755	0.773	0.776	0.778	0.764(4)	**2150.860**
M_ADA_A + RF	0.705	0.698	0.723	0.762	0.759	0.729(7)	1586.31
M_ADA_A + SVM	0.719	0.736	0.752	0.768	0.763	0.748(5)	**2263.326**
M_ADA_A_SMV + RF	0.759	0.762	0.771	0.778	0.775	**0.769(3)**	**61610.067**
M_ADA_A_SMV + SVM	0.763	0.771	0.775	0.775	0.781	**0.773(2)**	**61779.402**
MF_ADA + RF	0.639	0.665	0.668	0.673	0.679	0.665(12)	1394.256
MF_ADA + SVM	0.706	0.716	0.728	0.729	0.726	0.721(8)	1383.213
Proposed algorithm	**0.771**	**0.782**	**0.785**	**0.786**	**0.789**	**0.783(1)**	1613.834

Table 4
Comparison of different algorithms on dataset Dermatology.

Algorithms	F_a					F_{aa}	et_a
	Z = 10	Z = 20	Z = 30	Z = 40	Z = 50		
M1 + RF	0.727	0.758	0.800	0.786	0.778	0.770(13)	3.444
M1 + SVM	0.773	0.771	0.765	0.768	0.776	0.771(12)	4.053
SAMME + RF	0.775	0.869	0.905	0.925	0.936	0.882(6)	3.39
SAMME + SVM	0.831	0.855	0.846	0.853	0.866	0.850(7)	3.771
SAMMER + RF	0.782	0.789	0.793	0.826	0.858	0.810(9)	3.411
SAMMER + SVM	0.873	0.868	**0.960**	0.951	0.882	0.907(4)	3.81
M_ADA_A + RF	0.782	0.791	0.798	0.812	0.835	0.804(11)	3.468
M_ADA_A + SVM	0.869	0.886	0.912	0.909	0.889	0.893(5)	3.795
M_ADA_A_SMV + RF	0.938	**0.951**	0.955	**0.954**	0.946	**0.949(2)**	**117.501**
M_ADA_A_SMV + SVM	0.936	0.941	0.945	0.951	0.947	**0.944(3)**	**285.693**
MF_ADA + RF	0.792	0.806	0.812	0.808	0.806	0.805(10)	3.366
MF_ADA + SVM	0.812	0.818	0.833	0.856	0.856	0.835(8)	3.915
Proposed algorithm	**0.944**	0.949	0.955	0.952	**0.951**	**0.950(1)**	4.124

Table 5
Comparison of different algorithms on dataset Arrhythmia.

Algorithms	F_a					F_{aa}	et_a
	Z = 10	Z = 20	Z = 30	Z = 40	Z = 50		
M1 + RF	0.402	0.409	0.415	0.420	0.423	0.414(13)	53.928
M1 + SVM	0.458	0.451	0.453	0.462	0.438	0.452(8)	**79.023**
SAMME + RF	0.418	0.435	0.459	0.456	0.448	0.443(9)	52.626
SAMME + SVM	0.462	0.472	0.484	0.485	0.489	0.478(4)	72.648
SAMMER + RF	0.408	0.416	0.435	0.459	0.462	0.436(10)	52.206
SAMMER + SVM	0.454	0.468	0.476	0.477	0.483	0.472(5)	**120.822**
M_ADA_A + RF	0.401	0.407	0.427	0.422	0.435	0.418(12)	51.261
M_ADA_A + SVM	0.502	0.489	0.468	0.476	0.489	**0.485(3)**	54.261
M_ADA_A_SMV + RF	0.504	**0.510**	**0.503**	0.498	0.504	**0.504(2)**	**1748.742**
M_ADA_A_SMV + SVM	0.489	0.456	0.450	0.446	0.449	0.458(6)	**2675.634**
MF_ADA + RF	0.332	0.441	0.445	0.469	0.456	0.429(11)	51.726
MF_ADA + SVM	0.456	0.459	0.462	0.456	0.451	0.457(7)	51.15
Proposed algorithm	**0.506**	0.505	0.500	**0.503**	**0.512**	**0.505(1)**	73.290

Table 6
Comparison of different algorithms on dataset DrivFace.

Algorithms	F_a					F_{aa}	et_a
	Z = 10	Z = 20	Z = 30	Z = 40	Z = 50		
M1 + RF	0.765	0.765	0.768	0.771	0.778	0.769(13)	909.516
M1 + SVM	0.782	0.805	0.828	0.826	0.831	0.814(10)	1127.04
SAMME + RF	0.808	0.816	0.813	0.821	0.818	0.815(9)	922.722
SAMME + SVM	0.819	0.823	0.841	0.842	0.848	0.835(7)	1125.693
SAMMER + RF	0.832	0.841	0.852	0.855	0.838	0.844(6)	887.151
SAMMER + SVM	0.820	0.798	0.766	0.772	0.779	0.787(12)	**1410.759**
M_ADA_A + RF	0.835	0.845	0.851	0.856	0.858	0.849(5)	**1371.288**
M_ADA_A + SVM	0.831	0.856	0.862	**0.865**	0.861	**0.855(3)**	**1409.142**
M_ADA_A_SMV + RF	0.852	0.852	0.855	0.856	0.858	**0.856(2)**	**16572.207**
M_ADA_A_SMV + SVM	0.845	0.849	0.856	0.858	0.862	0.854(4)	**26479.662**
MF_ADA + RF	0.812	0.805	0.786	0.792	0.810	0.801(11)	841.5
MF_ADA + SVM	0.792	0.819	0.835	0.828	0.826	0.820(8)	901.929
Proposed algorithm	**0.858**	**0.861**	**0.866**	0.862	**0.868**	**0.863(1)**	1350.457

each dataset are denoted in bold. Obviously, when the datasets of DrivFace, Arrhythmia and Dermatology are used, we can see that M_ADA_A and M_ADA_A_SMV outputs the other algorithms like SAMME and SAMMER. The reason may be that these algorithms consider the AUC information when calculating the classifier weights thus are much more suitable for imbalanced datasets than the other algorithms. When comparing M_ADA_A_SMV with M_ADA_A, it is obvious that the performance of former algorithm is significantly better than that of the later one, showing that combining the results of multiple AdaBoost classifiers are very helpful in improving the classification accuracy. When compare the proposed algorithm with the other algorithms, as is reported in Tables 2-9, the proposed algorithm obtains the highest *F*a values for 25 times, showing that it outperforms the other algorithms with the probability of 62.5 % (25 of 40 cases). For example, when the datasets of Amazon and CNAE9 are used, the *F*a values of the proposed algorithm are all obviously higher than those of the other algorithms; when the datasets of Sonar, Arrhythmia, DrivFace and Gisette are used respectively, the proposed algorithm obtains the highest *F*a values for more than twice. When considering the *F*aa

Table 7
Comparison of different algorithms on dataset Splice.

Algorithms	F_a					F_{aa}	et_a
	Z = 10	Z = 20	Z = 30	Z = 40	Z = 50		
M1 + RF	0.912	0.887	0.938	0.909	0.949	0.919(8)	7.389
M1 + SVM	0.896	0.907	0.918	0.919	0.922	0.912(10)	9.537
SAMME + RF	0.906	0.912	0.931	0.936	0.934	0.924(5)	7.503
SAMME + SVM	0.886	0.911	0.916	0.919	0.925	0.911(11)	9.006
SAMMER + RF	0.898	0.906	0.918	0.938	0.944	0.921(7)	7.572
SAMMER + SVM	0.866	0.906	0.928	0.939	0.936	0.915(9)	7.443
M_ADA_A + RF	0.951	0.954	0.955	**0.959**	**0.964**	**0.957(1)**	7.434
M_ADA_A + SVM	0.892	0.923	0.926	0.936	0.939	0.923(6)	9.45
M_ADA_A_SMV + RF	**0.955**	**0.955**	**0.956**	0.958	0.959	**0.957(1)**	**162.444**
M_ADA_A_SMV + SVM	0.949	0.942	0.938	0.935	0.933	0.939(4)	**197.652**
MF_ADA + RF	0.762	0.793	0.808	0.825	0.839	0.805(12)	7.14
MF_ADA + SVM	0.748	0.768	0.779	0.817	0.815	0.785(13)	9.501
Proposed algorithm	0.952	0.952	0.954	0.954	0.957	**0.954(3)**	20.404

Table 8
Comparison of different algorithms on dataset CNAE9.

Algorithms	F_a					F_{aa}	et_a
	Z = 10	Z = 20	Z = 30	Z = 40	Z = 50		
M1 + RF	0.882	0.892	0.897	0.987	0.898	0.911(7)	266.919
M1 + SVM	0.875	0.879	0.879	0.886	0.882	0.880(12)	346.017
SAMME + RF	0.879	0.889	0.898	0.902	0.905	0.895(9)	273.018
SAMME + SVM	0.865	0.869	0.877	0.866	0.856	0.867(13)	322.341
SAMMER + RF	0.862	0.886	0.887	0.898	0.896	0.886(11)	268.02
SAMMER + SVM	0.899	0.892	0.886	0.883	0.882	0.888(10)	282.633
M_ADA_A + RF	0.932	0.923	0.908	0.905	0.902	0.914(6)	**662.466**
M_ADA_A + SVM	0.921	0.919	0.917	0.912	0.909	0.916(4)	**741.846**
M_ADA_A_SMV + RF	0.926	0.935	0.932	0.939	0.946	**0.936(2)**	**7148.919**
M_ADA_A_SMV + SVM	0.922	0.931	0.936	0.937	0.936	**0.932(3)**	**11545.611**
MF_ADA + RF	0.898	0.893	0.890	0.905	0.896	0.896(8)	261.099
MF_ADA + SVM	0.922	0.918	0.913	0.912	0.910	0.915(5)	274.527
Proposed algorithm	**0.940**	**0.943**	**0.945**	**0.945**	**0.951**	**0.945(1)**	417.293

Table 9
Comparison of different algorithms on dataset Gisette.

Algorithms	F_a					F_{aa}	et_a
	Z = 10	Z = 20	Z = 30	Z = 40	Z = 50		
M1 + RF	0.965	0.967	0.975	0.968	0.976	0.970(10)	5369.760
M1 + SVM	0.975	0.976	0.978	0.981	0.981	0.978(4)	6847.248
SAMME + RF	0.962	0.968	0.971	0.978	0.977	0.971(9)	5368.428
SAMME + SVM	0.975	0.974	0.977	0.979	0.973	0.975(7)	7039.539
SAMMER + RF	0.964	0.966	0.972	0.973	0.976	0.970(10)	5884.317
SAMMER + SVM	**0.983**	0.975	0.980	0.977	0.974	0.978(4)	8400.756
M_ADA_A + RF	0.945	0.958	0.956	0.955	0.961	0.955(13)	7927.392
M_ADA_A + SVM	0.946	0.951	0.960	0.966	0.976	0.960(12)	8215.272
M_ADA_A_SMV + RF	0.977	0.979	0.975	0.979	0.978	0.978(4)	**97690.62**
M_ADA_A_SMV + SVM	0.982	0.977	0.983	0.983	0.979	**0.981(2)**	**127205.499**
MF_ADA + RF	0.971	0.975	0.977	0.978	0.976	0.975(7)	5444.703
MF_ADA + SVM	0.981	**0.981**	0.983	0.979	0.979	**0.981(2)**	5839.911
Proposed algorithm	0.982	0.975	**0.985**	**0.985**	**0.981**	**0.982(1)**	12806.513

values, we find that the proposed algorithm achieves the highest *F*aa values for 7 times, with the highest value 0.982 on Gisette dataset. Moreover, in order to further validate the effectiveness of the proposed algorithm, we sort the *F*aa values of all algorithms in descending order with respect to each dataset and give their order numbers in the brackets in Tables 2-9. It can be easily seen that the proposed algorithm, M_ADA_A_SMV + RF and M_ADA_A_SMV + SVM are the top three best algorithms and they obtain the top three best results for 8, 6 and 4 times respectively, validating the improvement of our algorithm on classification accuracy when compared with typical AdaBoost based algorithms.

Further, in order to validate the execution speed of the proposed algorithm, as is shown in Tables 2-9, we obtain the average execution time (called *et*a, expressed in seconds) of the above combined algorithms on each dataset when the ratio of selected features ranges from 0.1 to 0.5 with a step of 0.1 and the iteration number Z ranges from 10 to 50 with a step of 10. For each dataset, the *et*a values which are higher than that of the proposed algorithm are denoted in bold. It can be seen from Tables 2-9 that, the M_ADA_A_SMV based algorithms (M_ADA_A_SMV + RF and M_ADA_A_SMV + SVM) are the slowest ones in all algorithms. This is due to the reason that M_ADA_A_SMV has to construct multiple

binary sub-learners in which each one is a AdaBoost algorithm and consists of Z base learners. According to sections 3.1 and 5.2, we know that the execution time of WNMF in our algorithm is affected by the sample number (M) and the feature number (N). The higher the values of M and N are, the higher the execution time of WNMF is. It can be easily seen that, when the datasets of Splice and Gisette are used, the *et*a values of the proposed algorithm are higher than that of the algorithms like M1 + RF, SAMME + RF and MF_ADA + RF but much lower than that of the M_ADA_A_SMV based algorithms. When the datasets of Sonar, Dermatology and Arrhythmia are used, the *et*a values of the proposed algorithm are similar with that of the other algorithms except for the M_ADA_A_SMV based algorithms. The reason is that the values of M and N in these datasets are both smaller than 500, deducing little effect on increasing the execution time of the proposed algorithm. Moreover, when the datasets of Amazon, DrivFace and CNAE9 are used, we observe that the proposed algorithm runs faster than at least four algorithms, showing its effectiveness on improving the classification accuracy while keeping good performance on execution speed.

6. Conclusions

An improved AdaBoost algorithm based on misclassified samples oriented feature selection and weighted non-negative matrix factorization is proposed in this paper. The main characteristics include: (1) As existing feature selections ignore the sample weight information, a misclassified samples oriented feature selection (MOFS) is proposed to improve the classification ability of AdaBoost on dealing with misclassified samples; (2) As the traditional AdaBoost algorithms ignore the part-based features of the training samples, the advantage of weighted non-negative matrix factorization (WNMF) on detecting the part-based features is utilized. Moreover, MOFS is combined with WNMF to both consider the explicit features and the part-based features of the training samples to improve the classification accuracy. (3) In order to distinguish the effects of different misclassified samples, the fine grained sample weight updating method and the macro F_1 measurement considered classifier weight calculation method are proposed. Experiments show the effectiveness of the proposed algorithm on improving the classification accuracy when different datasets are used. In the future, we will use two typical deep learning algorithms rather than traditional classifiers like RF and SVM to represent h_a and h_b. Moreover, as the parameters used in misclassification degree calculation process are set to be constants in this paper, we will carry out further research to learn the best values of these parameters to improve the classification performance of the proposed algorithm.

CRediT authorship contribution statement

Youwei Wang: Conceptualization, Methodology. **Lizhou Feng:** Visualization. **Jianming Zhu:** Supervision. **Yang Li:** Validation. **Fu Chen:** Investigation.

Data availability

Data will be made available on request.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (No. 61906220), the Ministry of Education of Humanities and Social Science Project (No. 19YJCZH178), National Social Science Foundation of China (No. 18CTJ008), the Natural Science Foundation of Tianjin Province (No. 18JCQNJC69600), the National Key R&D Program of China (2017YFB1400700) and the Emerging Interdisciplinary Project of CUFE.

References

[1] G.I. Webb, Z. Zheng, Multistrategy ensemble learning: reducing error by combining ensemble learning techniques, IEEE Trans. Knowl. Data Eng. 16 (2004) 980–991.
[2] M. Galar, A. Fernandez, E. Barrenechea, A review on ensembles for the class imbalance problem: bagging, boosting, and hybrid-based approaches, IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 42 (2012) 463–484.
[3] P.N. Thanh, M. Kappas, Comparison of random forest, k-Nearest Neighbor, and support vector machine classifiers for land cover classification using sentinel-2 imagery, Sensors 18 (2018).
[4] B. Sun, H. Chen, J. Wang, et al., Evolutionary under-sampling based bagging ensemble method for imbalanced data classification, Front. Comput. Sci. 12 (2018) 331–350.
[5] S. Hido, H. Kashima, Y. Takahashi, Roughly balanced bagging for imbalanced data, Stat. Anal. Data Min. 2 (2010) 412–426.
[6] R.E. Schapire, The strength of weak learnability, Machine Learning 5 (1990) 197–227.
[7] X. Li, L. Wang, E. Sung, AdaBoost with SVM-based component classifiers, Eng. Appl. Artif. Intell. 21 (2008) 785–795.
[8] M.M. Baig, M.M. Awais, E.S.M. El-Alfy, AdaBoost-based artificial neural network learning, Neurocomputing 248 (2017) 120–126.
[9] X. Yao, X.D. Wang, Y.X. Zhang, A Self-Adaption Ensemble Algorithm Based on Random Subspace and AdaBoost, Acta Electronica Sinica 41 (2013) 810–814.
[10] J.D. Wang, P. Li, R. Ran, A short-term photovoltaic power prediction model based on the gradient boost decision tree, Appl. Sci. 8 (2018) 689–703.
[11] C. Zhuo, J. Fu, Y. Cheng, Xgboost classifier for DDoS attack detection and analysis in SDN-based cloud, In: IEEE International Conference on Big Data and Smart Computing (BigComp), 2018.
[12] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the Thirteenth International Conference on Machine Learning, Italy, 1996, 148-156.
[13] E. Schapire, Robert, Improved boosting algorithms using confidence-rated predictions, Machine Learning 37 (1999) 297–336.
[14] J. Zhu, H. Zou, S. Rosset, Multi-class AdaBoost, Statistics and Its Interface 2 (2009) 349-360.
[15] X.W. Yang, Z. Ma, S. Yuan, Multi-class AdaBoost algorithm based on the adjusted weak classifier, J. Electron. Inf. Technol. 38 (2016) 373–380.
[16] Y. Freund, R.E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1997) 119–139.
[17] D.P. Solomatine, D.L. Shrestha, AdaBoost.RT: A boosting algorithm for regression problems, in: Proceedings of the Int Joint Conf on Neural Networks, Budapes, 2004, 1163-1168.
[18] B. Sun, S. Chen, J. Wang, A robust multi-class AdaBoost algorithm for mislabeled noisy data, Knowl.-Based Syst. 102 (2016) 87–102.
[19] D. Tang, L. Tang, R. Dai, MF-Adaboost: LDoS attack detection based on multi-features and improved AdaBoost, Fut. Generat. Comput. Syst. 106 (2020) 347–359.
[20] K.W. Li, G.Y. Zhou, J.N. Zhai, et al., Improved PSO_AdaBoost ensemble algorithm for imbalanced data, Sensors 19 (2019).
[21] Y. Zhou, T.A. Mazzuchi, S. Sarkani, M-AdaBoost-A based ensemble system for network intrusion detection, Expert Syst. Appl. 162 (2020) 113864.
[22] H. Guo, Y. Li, Y. Li, BPSO-AdaBoost-KNN ensemble learning algorithm for multi-class imbalanced data classification, Eng. Appl. Artif. Intell. 49 (2016) 176–193.
[23] B. Tang, H.B. He, GIR-based ensemble sampling approaches for imbalanced learning, Pattern Recogn. 71 (2017) 306–319.
[24] F. Muhlenbach, S. Lallich, D.A. Zighed, Identifying and handling mislabeled instances, J. Intell. Inf. Syst. 22 (2004) 89–109.
[25] R.A. Servedio, Smooth boosting and learning with malicious noise, J. Mach. Learn. Res. 4 (2004) 473–489.
[26] J. Cao, S. Kwong, R. Wang, A noise-detection based AdaBoost algorithm for mislabeled data, Pattern Recogn. 45 (2012) 4451–4465.
[27] Q. Yang, X. Wu, 10 challenging problems in data mining research, Int. J. Inf. Technol. Decis. Making 05 (2006) 597–604.
[28] W. Lu, Z. Li, J. Chu, Adaptive ensemble undersampling-boost: a novel learning framework for imbalanced data, J. Syst. Softw. 132 (2017) 272–282.
[29] N.V. Chawla, K.W. Bowyer, L.O. Hall, SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2011) 321–357.
[30] H. Han, W.Y. Wang, B.H. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: International Conference on Intelligent Computing, Springer, 2005, pp. 878-887.

[31] H. He, Y. Bai, E.A. Garcia, et al, ADASYN: adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IEEE, pp. 1322-1328.
[32] K. Li, X. Fang, J. Zhai, et al., An imbalanced data classification method driven by boundary samples-Boundary-Boost, in, in: Information Science and Control Engineering (ICISCE), 3rd International Conference on, 2016, pp. 194–199.
[33] B. Tang, H. He, Kernel, ADASYN: Kernel based adaptive synthetic data generation for imbalanced learning, IEEE Congress Evolut. Comput. (CEC) (2015).
[34] P. Kang, S. Cho, EUS SVMs: ensemble of under-sampled SVMs for data imbalance problems, in: Neural Information Processing, Springer, 2006, pp. 837-846.
[35] L. Bao, C. Juan, J. Li, et al., Boosted near-miss under-sampling on SVM ensembles for concept detection in large-scale imbalanced datasets, Neurocomputing 172 (2016) 198–206.
[36] Y. Tian, X. Wang, SVM ensemble method based on improved iteration process of AdaBoost algorithm, In: 29th Chinese Control And Decision Conference (CCDC), 2017.
[37] R. Kohavi, G.H. John, Wrappers for feature subset selection, Artif. Intell. 97 (1997) 273–324.
[38] Y. Yang, J.O. Pedersen, A comparative study on feature selection in text categorization, in: Proceedings of the Fourteenth International Conference on Machine Learning, 1997, pp. 412–420.
[39] J.M. Yang, Z.Y. Qu, Z.Y. Liu, Improved feature selection method considering the imbalance problem in text categorization, Sci. World J. (2014) 1–17.
[40] S.S.R. Mengle, N. Goharian, Ambiguity measure feature-selection algorithm, J. Am. Soc. Inform. Sci. Technol. 60 (2009) 1037–1050.
[41] Y. Wang, L. Feng, J. Zhu, Novel artificial bee colony based feature selection for filtering redundant information, Appl. Intell. (2017) 1–18.
[42] M. Mazini, B. Shirazi, I. Mahdavi, Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms, J. King Saud Univ. – Comput. Inf. Sci. 31 (2019) 541–553.
[43] D. Zhang, L. Zou, X. Zhou, Integrating feature selection and feature extraction methods with deep learning to predict clinical outcome of breast cancer, IEEE Access 6 (2018) 28936–28944.
[44] Y. Cao, J.C. Liu, Q.G. Miao, Improved behavior-based malware detection algorithm with AdaBoost, J. Xidian Univ. (Natural Science) 6 (2013) 116–124.
[45] W. Lee, C.H. Jun, J.S. Lee, Instance categorization by support vector machines to adjust weights in AdaBoost for imbalanced data classification, Inf. Sci. 381 (2017) 92–103.
[46] H. Xiao, Z. Xiao, Y. Wang, Ensemble classification based on supervised clustering for credit scoring, Appl. Soft Comput. 43 (2016) 73–86.
[47] S. Yang, L.F. Chen, T. Yan, An ensemble classification algorithm for convolutional neural network based on AdaBoost, in: 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS). IEEE Computer Society.
[48] M. Yousefi, M. Yousefi, R.P.M. Ferreira, Chaotic genetic algorithm and AdaBoost ensemble metamodeling approach for optimum resource planning in emergency departments, Artif. Intell. Med. 84 (2018) 23–33.
[49] X. Gao, C. Shan, C. Hu, An adaptive ensemble machine learning model for intrusion detection, IEEE Access 7 (2019) 82512–82521.
[50] Y.B. Chen, P. Dou, X.J. Yang, Improving land use/cover classification with a multiple classifier system using AdaBoost integration technique, Remote Sens. 9 (2017) 1055–1075.
[51] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (1999) 788–791.
[52] V.D. Blondel, N.D. Ho, P.V. Dooren, Weighted nonnegative matrix factorization and face feature extraction, Image Vis. Comput. (2007) 1–17.
[53] J. Kim, H. Park, Sparse nonnegative matrix factorization for clustering, Technical Report GT-CSE-08-01, Georgia Institute of Technology, 2008.
[54] L. H. Zhao, G. B. Zhuang, X. H. Xu, Facial expression recognition based on PCA and NMF, in: Proceedings of 7th World Congress on Intelligent Control and Automation, June 25-27, 2008, Chongqing, China.
[55] R. Dubroca, C. Junor, A. Souloumiac, Weighted NMF for High-Resolution Mass Spectrometry Analysis, EUSIPCO, August 27–31, Bucharest, Romania, 2012.
[56] J. Dougherty, R. Kohavi, M. Sahami, Supervised and UnSupervised Discretization of Continuous Features, Morgan Kaufmann Publishers, San Francisco, CA, 1995.
[57] X. Dai, N. Zhang, K. Zhang, J. Xiong, Weighted nonnegative matrix factorization for image inpainting and clustering, Int. J. Comput. Intell. Syst. 13 (2020) 734–743.
[58] L. Liang, L. Shan, F. Liu, Sparse envelope spectra for feature extraction of bearing faults based on NMF, Appl. Sci. 9 (2019) 1–22.
[59] A. Asuncion, D.J. Newman, UCI Machine Learning Repository, University of California, Department of Information and Computer Science, Irvine, CA, 2007.
[60] B. Amarnath, S. Balamurugan, A. Alias, Review on feature selection techniques and its impact for effective data classification using UCI machine learning repository dataset, J. Eng. Sci. Technol. 11 (2016) 1639–1646.
[61] Y.W. Wang, L.Z. Feng, et al., Two-step based feature selection method for filtering redundant information, J. Intell. Fuzzy Syst. 33 (2017) 2059–2073.
[62] L.M. Abualigah, A.T. Khader, Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering, J. Supercomput. 73 (2017) 4773–4795.

Youwei Wang, Associate professor of School of Information, Central University of Finance and Economics, Beijing, China. He received the Doctor degree in computer science and technology college from Jilin University of China in 2015. He received the master degree from Jilin University of China in 2011. His current research interests include data mining, deep learning, social network, etc.

Lizhou Feng, Associate professor of School of statistics, Tianjin University of Finance and Economics, Tianjin, China. She received the Doctor degree in computer science and technology college from Jilin University of China in 2015. She received the master degree from Jilin University of China in 2011. Her current research interests include data mining, text classification, Web Intelligence, etc.

Jianming Zhu, Professor of School of Information, Central University of Finance and Economics, Beijing, China. He is now a senior member of China Computer Society and also a member of the Committee on Information Security. He is a member of the special committee of the system and also a senior member of the China Communications Society (China Communications Association). More than 70 papers were published in international journals and conferences on Information Security.

Yang Li, Associate professor of School of Information, Central University of Finance and Economics, Beijing, China. He is now a member of China Computer Society. His current research interests include data mining, security protocol analysis, Fintech, etc.

Fu Chen, Professor of School of Information, Central University of Finance and Economics, Beijing, China. He is now a senior member of China Computer Society. He is a member of Internet Specialized Committee. His current research interests include data mining, IoT security and protocol analysis, social network, etc.