ABSTRACT Adaboost is a typical ensemble learning algorithm and has been studied and widely used in classification tasks. In order to improve the classification performance of existing Adaboost algorithms effectively, a noise confidence degree and weighted feature selection based Adaboost algorithm (called NW_Ada) is proposed. Firstly, in order to decrease the impact of sample set density on noise detection results, the conceptions of clustering degree and deviated degree are introduced, and a new method of evaluating the noise confidence is proposed. Then, based on the traditional feature selections of filters, a weighted feature selection method is proposed to select the features which can effectively distinguish the samples those are misclassified. Finally, based on the traditional error rate calculation method, the category recall based classifier error rate calculation method is proposed to solve the problem that the traditional methods ignore the distribution of misclassified samples when dealing with the unbalanced datasets. The experimental results show that the proposed method comprehensively considers the influences of sample density, sample weight and dataset size on classification results, and obtains significant improvement on classification performance compared to traditional Adaboost algorithms when different datasets especially the unbalanced datasets are used.

INDEX TERMS Ensemble learning, feature selection, error rate, classifier weight, sample density.

I. INTRODUCTION
Ensemble learning which can significantly improve the accuracy and generalization performance of learning systems, has received more and more attention in the field of pattern recognition and machine learning [1]–[3]. Traditional ensemble learning algorithms mainly includes two types: bagging and boosting [4]. Bagging algorithms randomly select the training samples of a weak classifier in each iteration, and simply combine the results of multiple weak classifiers to improve the classification stability. Typical bagging algorithms include: random forest [5], BEBS [6], RB Bagging [7], etc. Boosting algorithms emphasize the differences of weak classifiers, and improve the classification ability on misclassified samples by continuously adjusting the sample weights. Typical boosting algorithms include: Adaboost [8]–[10], gradient boosting decision tree (GBDT) [11], XGboost [12], etc. Compared with the bagging algorithms, boosting algorithms can effectively reduce the sample classification bias and improve the classification accuracy, thus has attracted wide attention in recent years.

Among many boosting based ensemble learning algorithms, AdaBoost has become one of the most popular algorithms because of its simple implementation and strong adaptability [8]–[10]. Adaboost is proposed to solve the dichotomy problem by Freund and Schapire [13]. This algorithm adaptively changes the training sample weights according to the classification result of the weak classifier in each
iteration, and obtains a new classifier by focusing on the samples which are misclassified by the weak classifier, thus promotes the weak classifier to a strong classifier. Based on the Adaboost algorithm, Adaboost.M1 [14], Adaboost.M2 [13], Adaboost.MH [15] and SAMME [16] are proposed to solve the multi-classification problems. However, the weak classifier accuracies of these algorithms are low and cannot effectively improve the performance of the final strong classifiers. On this basis, the SAMME.R algorithm is proposed to ensure the accuracy of the weak classifier by adding the limits on it [17]. Further, Freund and Schapire extended the AdaBoost algorithm to the regression field and introduced the AdaBoost.R algorithm [18]. Solomatine improved the AdaBoost.R algorithm and proposed the AdaBoost.RT algorithm [19] by introducing a threshold constant, transforming the regression problem into a simple two-class problem. In recent years, as the increasing of diversification and complexity of network data, the Adaboost algorithms are widely used in signal detection [20], link detection [21], time series prediction [22], image classification [23] and other fields.

Based on the existing Adaboost algorithms, an improved Adaboost algorithm based on noise confidence evaluation and weighted feature selection is proposed by considering the influence of sample densities, sample weights and category sizes. The main contributions of this article include: 1) The influence of sample density is considered in noise detection, and a noise confidence degree based new method is proposed to improve the accuracy of noise detection by introducing the conceptions of social degree and deviated degree. 2) The dynamic changes of sample weights are studied, and a weighted feature selection based on traditional filters is proposed to improve the classification ability of AdaBoost on dealing with the misclassified samples. 3) Because the traditional classifier error rate calculation method ignores the category information of the misclassified samples, a category recall based error rate calculation method is proposed to reduce the impact of category sizes.

The rest of this article is organized as follows. Section II introduces the related work of Adaboost algorithms. Section III gives the execution process of the traditional Adaboost algorithm. Section IV gives the proposed noise confidence degree and weighted feature selection based Adaboost algorithm. Section V gives the experimental results and analysis. Section VI concludes the whole paper.

II. RELATED WORK
Sample preprocessing and classifier enhancing are two main aspects of the research on Adaboost algorithms. In terms of sample preprocessing, existing algorithms generally select the representative samples for training by oversampling [24], undersampling [25], etc. By considering the distribution characteristics of the samples, the noise samples are removed so as to ensure the performance of Adaboost algorithms. In terms of classifier enhancing, most existing algorithms focus on the aspects of feature selection, weak classifier weight calculation and weak classifier combination, aiming at improving

Algorithm 1 Traditional Adaboost Algorithm

**Input:** training sample set $S$, label set $Y$, weak classifier $h$, number of iterations $T$

**Output:** the final classifier $H$

1. for $i = 1:1:M$
2. initialize the weight of sample $s_i$: $D_t(i) = 1/M$;
3. end for
4. for $t = 1:1:T$
5. select a sample subset $X$ from the training set $S$, train $X$ using $h$ to obtain a weak classifier $h_t$ and calculate the weighted error rate $e_t$:
   \[
   e_t = \sum_{s_i \in X, h_t(s_i) \neq Y(s_i)} D_t(i)
   \]
   where $h_t(s_i)$ is the predicted label of $s_i$ using classifier $h_t$, $Y(s_i)$ is the real label of $s_i$.
6. calculate the classifier weight of $h_t$:
   \[
   \alpha_t = \frac{1}{2} \log \left( \frac{1 - e_t}{e_t} \right)
   \]
7. update the weight of each sample in $S$:
8. for $i = 1:1:M$
9. $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & h_t(s_i) = y_i \\ e^{\alpha_t} & h_t(s_i) \neq y_i \end{cases}$
10. end for
11. get the final strong classifier $H$: for sample $s$, its class label $H(s)$ is
   \[
   H(s) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(s) \right)
   \]

the training efficiency, classification accuracy and generalization ability of the final strong classifier. Typical algorithms of sample preprocessing and weak classifier enhancing are given as follows:

A. SAMPLE PREPROCESSING
Cao et al. proposed an improved Adaboost algorithm by introducing a K-nearest neighbor (KNN) based noise detecting method in the sample preprocessing process. This algorithm detects the noise samples by counting the predicted labels of one sample’s K-nearest neighbor samples, and increases the weights of the misclassified noise samples in Adaboost algorithm to improve the classification
performance [26]. As the noise samples may reduce the generalization performance and robustness of the algorithm, Sun et al. combined SAMME and Cao’s algorithm and proposed an improved Adaboost algorithm for multi-classification tasks. This algorithm first divides the training samples into noise samples and non-noise samples. If the non-noise samples are misclassified, the sample weights are increased. If the noise samples are misclassified, the sample weights are set to 0, avoiding the effect of noise samples on classification performances [27].

In order to reduce the sizes of the training sets and improve the efficiency of Adaboost algorithms, Lu et al. proposed an improved Adaboost algorithm (EUSBoost) by using dynamic integrated under-sampling in the preprocessing process. This algorithm avoids the influence of sample set imbalance on classification results by applying non-returning undersampling method, and maximizes the inter-class variance adaptively to determine the classification threshold of EUSBoost [28]. For solving the problem of sample size imbalance, Wu et al. applied the undersampling method to the sample set by clustering the largest category into several clusters. This method avoids the blindness of traditional randomly undersampling methods, and balances the sizes of different categories while retaining the most useful information of the original sample set [29]. For selecting the most representative samples, Lee et al. combined the positional factors of the samples, and divided all samples into four categories according to their positions, improving the classification accuracy of Adaboost algorithm when dealing with unbalanced sample sets [30].

B. CLASSIFIER ENHANCING

Yao et al. used Particle Swarm Optimization (PSO) algorithm to find the optimal feature weight distribution that minimizes the classification error rate of Adaboost. Based on the optimal feature weight distribution, the features are sampled randomly to generate a feature subspace and then applied to the training process of Adaboost, increasing the accuracy of the weak classifiers [10]. Guo et al. proposed an ensemble learning algorithm by combining Binary Particle Swarm Optimization (BPSO), Adaboost and KNN. This algorithm improves the stability of Adaboost on extracting the key features from unbalanced datasets [31]. Cao et al. proposed an anti-noise loss function to limit the weight unrestricted expansion of the samples those cannot always be correctly classified, preventing the algorithm from over-focusing on noises or samples with singular values [32]. Sun et al. designed a new multi-class loss function, and optimized the function to obtain the best weight of the weak classifier, effectively improved the classification accuracy [27]. To ensure the accuracy and the difference of the weak classifiers, Xiao et al. proposed an adaptive ensemble algorithm based on the algorithms of clustering and Adaboost. Firstly, this algorithm uses the clustering algorithm to divide the training samples into multiple clusters. Then, the clusters are used to obtain the trained weak classifiers and the weight based voting strategy is used to determine the final result of the classifiers [33]. Although the traditional weak classifier is easy to train, it cannot obtain high classification accuracy. For solving this problem, Yang et al. proposed the Convolutional Neural Networks (CNN) based Adaboost algorithm (ACNN). ACNN gives the weights of the weak classifiers according to the error rates of the pre-training stage, and dynamically adjusts the sample weights and learning coefficients according to the error rate of each iteration [34]. Yousefi et al. used the Adaptive Network-based Fuzzy Inference System (ANFIS), the Feed Forward Neural Network (FFNN), and the Recurrent Neural Network (RNN) as weak classifiers, improving the classification accuracy of the Adaboost algorithm effectively [35]. Chen et al. proposed an improved Adaboost algorithm by combining multiple classifiers and used it to analyze the land use problem. This algorithm determines the weights of Support Vector Machine (SVM), C4.5 decision tree and Artificial Neural Network (ANN) through experiments, and combines these weak classifiers to obtain a strong classifier which has better classification performance [36].

However, the existing Adaboost based algorithms still face the following problems: 1) Many noise detection algorithms use the KNN method to evaluate the differences between the label of the sample to be detected and the labels of its neighborhood samples, resulting in the problem that the evaluation result is seriously affected by the category which has samples with high density; 2) The selected feature subset has a great influence on the classification result, but most of the existing feature selection methods ignore the influence of the changing sample weights in each iteration, deducing that the features contained in the misclassified samples cannot be selected; 3) The existing classifier error rate calculation method only considers the weight information of the misclassified samples and ignores their real category information, thus the weights of weak classifiers are probably inaccurate when dealing with the unbalanced sample set. For solving the above problems, this article studies from the aspects of noise detection, feature selection and classifier error rate calculation, and proposes an improved Adaboost algorithm for classification based on noise confidence degree and weighted feature selection.

III. TRADITIONAL ADABoost ALGORITHM

The traditional Adaboost algorithm trains multiple weak classifiers with respect to different training sample sets, and combines these classifiers to form a strong classifier [37]. Given a training sample set: $S = \{s_1, s_2, \ldots, s_M\}$, the label set of $S$: $Y = \{y_i | y_i \in \{-1, +1\}\}$, the traditional Adaboost algorithm is executed as follows [13], [18]:

IV. THE PROPOSED ALGORITHM

A. NOISE CONFIDENCE EVALUATION BASED ON SOCIAL DEGREE AND DEVIATED DEGREE

Reference [27] finds a noise according to whether a sample’s neighbors are misclassified. However, this noise
The detection method depends on the classification accuracy of the classifier, thus its computational complexity is determined by the computational complexity of the weak classifier. Reference [38] determines whether a sample is a noise based on the probability that the sample’s label is different with those of its neighbors. This method performs well in specific cases, but ignores the fact that the sample densities of the categories are always different, deducing the problem that the boundary samples maybe classified into noises.

As is shown in Fig.1, it is assumed that the classification boundary of categories $C_1$ and $C_2$ is $L_1$. If the number of neighbor samples is set to $k = 5$, then from [38] we know that samples $s$ and $s_2$ will be classified into noises. When $s_2$ is filtered, there is no changes on the classification boundary of $C_1$ and $C_2$. While, when $s$ is filtered, the corresponding classification boundary will become $L_2$, deducing that the sample $s_1$ is misclassified into $C_2$. Obviously, as sample $s$ plays a key role in determining the classification boundary of the sample set, it cannot be filtered as a noise directly. Therefore, we conclude that the method of [38] is insufficient as it just uses the information of label differences between a sample and its neighbors when detecting the noises.

Consequently, this article introduces the conceptions of social degree and deviated degree based on [38], and realizes noise detection by considering not only the label similarity between a sample and its neighbor samples but also the deviated degree information of the sample with respect to the categories. The details are given as follows:

**Definition 1 (Social Degree):** According to [38], given a sample $s$, the proportion of the samples which are the neighbors of $s$ and have the same label with $s$ is defined as $s$’s social degree $g(s)$, which is defined as:

$$g(s) = \frac{\sum_{s_i \in \text{Neig}(s, k)} \text{C}(s) \equiv \text{C}(s_i)}{k}$$  \hspace{1cm} (6)

where $\text{C}(s)$ represents the category label of $s$, and $\text{Neig}(s, k)$ represents the $k$-nearest neighbor samples of $s$. Obviously, $g(s) \in [0, 1]$, the lower $g(s)$ is, the lower the label similarity between $s$ and its neighbors, and the more likely $s$ is a noise sample.

**Definition 2 (Deviated Degree):** The extent that sample $s$ deviates from the samples of category $c$ is denoted as the deviated degree of $s$ for $c$, which is denoted as $p(s, c)$ and defined as follows:

$$p(s, c) = \exp \left( -\frac{d(s, k, c)}{\sum_{s_j \in \text{Neig}(s, k, c)} d(s_j, k, c) / k} \right)$$  \hspace{1cm} (7)

where $\exp$ is the e-based exponential function, $\text{Neig}(s, k, c)$ represents the $k$ nearest neighbor samples of $s$ in category $c$. As is shown in Fig.2, $d(s, k, c) = \sum_{s_j \in \text{Neig}(s, k, c)} d(s_j, s) / k$, where $d(s_j, s)$ represents the Euclidean distance of the samples $s_j$ and $s$. We know from formula (7) that, if $p(s, c) \in (0, 1)$, the higher $p(s, c)$ is, the closer from $s$ to the samples of $c$ is.

**Definition 3 (Noise Confidence Degree):** Assuming that the social degree of sample $s$ is $g(s)$, the deviated degree of $s$ with respect to $c$ is $p(s, c)$, then the noise confidence degree of $s$ is denoted as $\eta(s)$ and defined as follows:

$$\eta(s) = g(s) \times \exp \left( \frac{\max_{c \in \text{C} \setminus \text{C}(s)} p(s, c)}{p(s, \text{C}(s))} \right)$$  \hspace{1cm} (8)

It can be seen from formula (8) that, if a sample $s$ has a lower social degree value, a lower deviated degree value with respect to its category $\text{C}(s)$ and higher deviated degree values with respect to the other categories, it will output a lower $\eta(s)$ value and is likely to be a noise sample. For the two samples of $s$ and $s_2$ in Fig.1, there exists $g(s) = g(s_2)$, and it can be seen that we cannot determine whether $s$ and $s_2$ are noises only by using the social degree information. Further, we notice that the sample $s$ is at the boundary of the categories $C_1$ and $C_2$, and there exists $p(s, C_1) > p(s, C_2)$. While, sample $s_2$ is inside category $C_2$, and there exists $p(s_2, C_1) < p(s_2, C_2)$. Therefore, we have $\eta(s_2) < \eta(s)$ according to formula (8), and it is consistent with the analysis that $s_2$ is more likely to be a noise than $s$.

Further, the execution process of the proposed noise detection algorithm based on noise confidence degree is given as follows:

**FIGURE 1. Illustration of the problem in [38].**

**FIGURE 2. Illustration of deviated degree conception.**
The proposed noise confidence calculation method is based on confidence degree, noise detection algorithm, and the social degree of each category. We remove the constant coefficient $k$ from formula (9): $T_1 = M \times \left( MN + M \log M + \sum_{c_i} m_i \log m_i + (k + k^2) \times L \right) \quad (9)$

where $N$ is the feature number, $m_i$ is the sample number in category $c_i$, and $L$ is the number of categories. For ease of computation, we remove the constant coefficient $k$ from formula (9):

$T_1 = M \times \left( MN + M \log M + \sum_{c_i} m_i \log m_i + L \right) \quad (10)$

As there exists:

$M \log M = \left( \log m_1 \right) M$

we have:

$= m_1 \log M + m_2 \log M + \cdots + m_L \log M$

where $m_i$ is the sample number in category $c_i$. From Algorithm 2, we know that the time complexity of the proposed noise confidence calculation method is:

$T_1 = M \times (MN + 2M \log M + L)$ \quad (11)

Further, we remove the constant coefficients of formula (11), then we get:

$T_1 < M^2 N + M^2 \log M + M L$. Similarly, the time complexity of noise detection method in [27] is:

$T_2 = M^2 N^2 + M^2 \log M \quad (12)$

Generally, there exists $L \ll M$ and $L \ll N$, then we have $T_1 < T_2$, deducing that the time complexity of the proposed noise detection method based on social degree and deviated degree is lower than that of the method in [27].

### B. Weighted Feature Selection Method Based on Filters

For the multi-label based Adaboost algorithms, the features used by each weak classifier will directly affect the classification result. The method of [10] introduced the PSO optimization to obtain a random feature subspace, and trained Adaboost classifier through updated sample weights and feature weights. However, this method has two problems: (1) It treats all samples equally and ignores the influence of sample weights on the feature selection results; (2) PSO optimization is time consuming and easy to fall into local extremum, which in turn decreases the execution effectiveness of feature selection process. Moreover, most of the traditional Adaboost algorithms do not perform the feature selection process or just use the traditional feature selections (like CMFS [39], IG [40], CHI [41] and IMGI [42]) to obtain the best features [13], [16], [18], [27], deducing low classification accuracy as the influences of the changing sample weights are ignored.

We now take an example to show the problem of traditional feature selections. Table 1 gives a set of samples $S = \{s_1, s_2, s_3, s_4\}$, the corresponding real labels of these samples $\{c_1, c_1, c_2, c_3\}$, a set of features $F = \{f_1, f_2, f_3, f_4, f_5\}$, a set of category labels $C = \{c_1, c_2, c_3\}$ and the number of iterations $T = 4$. When CMFS is used, the CMFS values of the features in $F$ are $\{1, 1, 1, 1, 1\}$. Suppose the number of selected features is set to $ns = 2$, then $f_1$ and $f_2$ are selected to form the set of selected features $FS = \{f_1, f_2\}$. As is shown in Table 1, when the weak classifier of SVM [35] is used, the predicted labels of the samples are $\{c_1, c_1, c_3, c_3\}$ when $t = 1$. Obviously, the label of $s_3$ is misclassified into $c_3$ as the feature $f_4$ is not contained in $FS$. As the traditional Adaboost algorithms do not consider the sample weights in each iteration, the CMFS values of the features in $F$ will not change and the selected features are always be $\{f_1, f_2\}$ in the following iterations, deducing the problem that sample $s_3$ is misclassified by all weak classifiers. Therefore, we know that $s_3$ will be misclassified by the final strong classifier.

**Table 1. Sample information and the predicted labels when CMFS method is used.**

| Sample | Representation | Real Label | Predicted Label($t=1$) |
|--------|----------------|------------|-----------------------|
| $s_1$  | 1, 1, 1, 0, 0  | $c_1$      | $c_1$                |
| $s_2$  | 1, 1, 0, 0, 0  | $c_1$      | $c_1$                |
| $s_3$  | 0, 0, 0, 1, 0  | $c_2$      | $c_3$                |
| $s_4$  | 0, 0, 0, 0, 1  | $c_3$      | $c_3$                |

![Algorithm 2](https://example.com/algorithm2.jpg)

**Algorithm 2.** Noise Detection Algorithm Based on Noise Confidence Degree

**Input:** training sample set $ts = \{s_i\}$ ($1 \leq i \leq M$), label set $Y = \{y_i\}$ ($1 \leq i \leq M$), threshold of noise confidence degree $\theta$. **Output:** noise sample set ($ns$).

1. for each $s_i$ in $ts$
   // calculate the social degree of $s_i$:
2. obtain the $k$-nearest neighbor sample set of $s_i$ and denote it as $Neig(s_i, k)$.
3. calculate the social degree of $s_i$ according to formula (6) and denote it as $g(s_i)$.
   // calculate the deviated degree of $s_i$ with respect to each category:
4. for each category $c$ in $ts$
5. obtain the $k$-nearest neighbor sample set of $s_i$ in $c$ and denote it as $Neig(s_i, k, c)$.
6. calculate the deviated degree $p(s_i, c)$ using formula (7).
7. end for
8. calculate the noise confidence degree of $s_i$ using formula (8).
9. if $\eta(s_i) < \theta$
10. $ns \leftarrow ns \cup \{s_i\}$.
11. end if
12. end for
Based on the traditional feature selections of filters, this article proposes a filter-based weighted feature selection method. This method considers two aspects: (1) if a feature outputs a high score obtained by a filter, it has high discriminative ability and should be selected; (2) if a feature can distinguish the samples which have high sample weights, it is helpful for dealing with the misclassified samples and should be selected. On this basis, given a feature \( f \) and an existing filter based feature selection method FFS, the proposed weighted feature selection method FFSW is expressed as follows:

\[
FFSW (f) = FFS (f) \times \sum_{k \in ts} \left( \frac{p (y_i | x_{i,f}) \times d_i}{p (y_i | x_{i,f})} \right)
\]  

where \( s_i \) is a sample in the training set, \( x_{i,f} \) is the representation of \( s_i \) using feature \( f \), \( y_i \) is the real label of \( s_i \), \( d_i \) is the weight of \( s_i \), \( FFS(f) \) is the output score of \( f \) using FFS. As is shown in formula (14), \( p(y_i | x_{i,f}) \) is the probability that \( x_{i,f} \) is correctly classified, which can be obtained according to Bayes theorem:

\[
p (y_i | x_{i,f}) = \frac{p (y_i) \times p (x_{i,f} | y_i)}{p (x_{i,f})}
\]

where \( p(y_i) \) is the probability that a sample occurs in category \( y_i \), \( p(x_{i,f} | y_i) \) is the probability that \( x_{i,f} \) occurs in category \( y_i \), \( p(x_{i,f}) \) is the probability that \( x_{i,f} \) occurs in \( ts \). Taking the CMFS method as an instance of FFS, the improved weighted feature selection method based on CMFS (denoted as CMFSW) is demonstrated as follows:

It is known from Algorithm 3 that the time complexity of CMFSW and CMFS are both: \( T = NM + NL + N \log N \), showing that the former does not bring obvious computational overhead while considering the sample weight information in feature selection process.

To validate the efficient of proposed feature selection method on dealing with the misclassified samples, we apply the CMFSW algorithm to the samples of Table 1 and the corresponding results are shown in Table 2 and Table 3.

**Algorithm 3 The Executing Process of CMFSW**

**Input:** training sample set \( ts = \{s_i\} = \{x_i, y_i\} (1 \leq i \leq M) \), \( x_i = \{x_j\} (1 \leq j \leq N) \), label set of the samples \( Y = \{y_i\} (1 \leq i \leq M) \), label set \( C = \{c_i\} (1 \leq i \leq L) \), sample weight set \( D = \{d_i\} (1 \leq i \leq M) \), \( d_i \) is the weight of sample \( s_i \), entire feature set \( F = \{f_i\} (1 \leq i \leq N) \), number of selected features \( N_1 \).

**Output:** final selected features subset \( fs \).

1. for \( i = 1: 1: N \) 
   2. calculate the number \( (nfi) \) that the feature \( f_i \) occurs in \( ts \):

\[
\{ \begin{align*}
nfi & = \sum_{s_j \in ts} \text{exist}(f_i, s_j) \\
\text{exist}(f_i, s_j) & = \begin{cases} 1, & \text{if } f_i \text{ occurs } ins_j \\ 0, & \text{else} \end{cases} 
\end{align*} \]

15. according to the CMFSW values, we sort all features of \( F \) in descending order, and select the top \( N_1 \) features to obtain the final selected feature subset \( fs \).

**TABLE 2. Feature weights in each iteration when CMFSW is used.**

| features | feature weights in each iteration |
|----------|----------------------------------|
|          | \( t=1 \) | \( t=2 \) | \( t=3 \) | \( t=4 \) |
| \( f_1 \) | 0.750    | 0.684    | 0.684    | 0.684    |
| \( f_2 \) | 0.750    | 0.684    | 0.684    | 0.684    |
| \( f_3 \) | 0.250    | 0.228    | 0.228    | 0.228    |
| \( f_4 \) | 0.667    | 0.754    | 0.754    | 0.754    |
| \( f_5 \) | 0.667    | 0.579    | 0.579    | 0.579    |

From these tables we know that, once \( s_4 \) is misclassified when \( t = 1 \), the sample weight of \( s_4 \) and the CMFSW values of the features contained in \( s_4 \) will increase in the next iteration, deducing that the feature \( f_4 \) is selected by using Algorithm 3 when \( t = 2 \). Further, we know from Table 3 that all samples are correctly classified by the second weak classifier \( h_2 \), and the sample weights and the classification results of the following weak classifiers remain unchanged. Therefore, we know that all samples are correctly classified by the final strong classifier \( H \), illustrating the advantage of proposed feature selection method in dealing with the misclassified samples when comparing with the traditional feature selection methods.
TABLE 3. Sample weights and predicted labels when CMFSW is used.

| sample | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $H$ |
|--------|-------|-------|-------|-------|-----|
| sample weight | labels | sample weight | labels | sample weight | labels | sample weight | labels | sample weight | labels | $H$ |
| $s_1$ | 0.250 | $c_1$ | 0.185 | $c_1$ | 0.185 | $c_1$ | 0.185 | $c_1$ | $c_1$ | $c_1$ |
| $s_2$ | 0.250 | $c_1$ | 0.185 | $c_1$ | 0.185 | $c_1$ | 0.185 | $c_1$ | $c_1$ | $c_1$ |
| $s_3$ | 0.250 | $c_3$ | 0.445 | $c_2$ | 0.445 | $c_2$ | 0.445 | $c_2$ | $c_2$ | $c_2$ |
| $s_4$ | 0.250 | $c_3$ | 0.185 | $c_1$ | 0.185 | $c_3$ | 0.185 | $c_3$ | $c_3$ | $c_3$ |

C. CATEGORY RECALL BASED CLASSIFIER ERROR RATE CALCULATION

Suppose the training set has $L$ categories and is denoted as $ts = \{ s_1, s_2, \ldots, s_M \}$, the label set of the samples is $Y = \{ y_1, y_2, \ldots, y_M \}$, the category set is $C = \{ c_i \} (1 \leq i \leq L)$, the set of weak classifiers is $hs = \{ h_1, h_2, \ldots, h_T \}$. According to SAMME algorithm [16] we know that the predicted label of sample $s$ can be obtained using the following formula:

$$H(s) = \arg \max_{c_j \in C} \sum_{i=1}^{T} \alpha_i \times h_{ij}(s)$$  \hspace{1cm} (19)

where

$$\alpha_i = \ln \left( \frac{1 - \varepsilon_i}{\varepsilon_i} \right) + \ln (L - 1)$$

$$h_{ij}(s) = \begin{cases} 1, & \text{if } h_i(s) = c_j \\ 0, & \text{else} \end{cases}$$

$$\varepsilon_i = \sum_{s' \in ts, h_i(s') \neq Y(s')} D_i(s')$$  \hspace{1cm} (20)

In formula (20), $h_i(s)$ is the predicted label of $s$ using classifier $h_i$. $Y(s')$ is the real label of sample $s'$, $\varepsilon_i$ is the error rate of classifier $h_i$ using the traditional error rate calculation method in Adaboost based algorithms (like the algorithms of [10], [13], [16] and [17]), $D_i(s')$ is the weight of sample $s'$ at the $i$th iteration. It can be seen that traditional error rate calculation method only considers the weights of misclassified samples, ignoring the distribution characteristics of misclassified samples in different categories, deducing that it is difficult to deal with the unbalanced datasets. As is shown in Table 4, assume that there are 3 samples which are misclassified by two weak classifiers $h_1$ and $h_2$ in the unbalanced training set $ts$ which has two categories of $c_1$ and $c_2$. When the traditional error rate calculation method is used, we have $\varepsilon_1 = \varepsilon_2 = 3/8$ and $\alpha_1 = \alpha_2 = \ln(5/3) \approx 0.511$.

It can be seen from the values of $h_1(s)$ that, although $h_1$ performs well on $c_1$ category, it misclassifies all samples of $c_2$ category. Therefore, $h_1$ has insufficient classification ability on dealing with category $c_2$, and its corresponding classifier weight $\alpha_1$ should be decreased. Moreover, we know from the values of $h_2(s)$ that $h_2$ can correctly classify most of the samples in $c_1$ and $c_2$, thus its corresponding classifier weight $\alpha_2$ should be greater than that of $h_1$.

TABLE 4. The problem of error rate calculation in traditional adaboost algorithms.

| sample set | $D(s)$ | $Y(s)$ | $h_1(s)$ | $h_2(s)$ |
|------------|--------|--------|----------|----------|
| $s_1$      | 0.125  | $c_1$  | $c_1$    | $c_1$    |
| $s_2$      | 0.125  | $c_1$  | $c_1$    | $c_2$    |
| $s_3$      | 0.125  | $c_1$  | $c_1$    | $c_1$    |
| $s_4$      | 0.125  | $c_1$  | $c_1$    | $c_1$    |
| $s_5$      | 0.125  | $c_1$  | $c_1$    | $c_1$    |
| $s_6$      | 0.125  | $c_2$  | $c_1$    | $c_2$    |
| $s_7$      | 0.125  | $c_2$  | $c_1$    | $c_1$    |
| $s_8$      | 0.125  | $c_2$  | $c_1$    | $c_2$    |

For solving the above problem of traditional error rate calculation method, this article comprehensively considers the distribution of misclassified samples in different categories, and proposes a new error rate calculation method based on category recall information to deal with the unbalanced datasets. Given the weak classifier $h_i$, the error rate of $h_i$ is denoted as $\delta_i$ and obtained as follows:

$$\delta_i = \varepsilon_i \times r_i$$

$$r_i = \frac{1}{L} \sum_{c_j \in C} \frac{n_{ij}}{n_j}$$  \hspace{1cm} (21)

where $r_i$ is the average value of the category recall values of all categories, $\varepsilon_i$ is the sum of the weights of the samples which are misclassified by classifier $h_i$, $n_j$ is the number of samples in category $c_j$, and $n_{ij}$ is the number of samples which are misclassified into $c_j$. Obviously, the proposed error rate calculation method based on category recall information treats all categories equally and avoids the problem that a weak classifier which misclassifies most of the samples in a small category be assigned a high weight. According to formula (21) we know that, the error rates of $h_1$ and $h_2$ are $\delta_1 = 3/8 \times 1/2 = 3/16 \approx 0.188$ and $\delta_2 = 3/8 \times 11/30 = 11/80 \approx 0.138$, respectively. Further, we combine formula (20) and obtain $\alpha_1 = \ln(13/3) \approx 1.466 < \alpha_2 = \ln(69/11) \approx 1.792$, which indicates that the accuracy of the proposed error rate calculation method based on category recall is higher compared to that of traditional error rate calculation method when dealing with the unbalanced datasets.
Algorithm 4 Improved Adaboost Algorithm Based on Noise Confidence Degree and Weighted Feature Selection (NW_Ada)

**Input:** sample set: \( TS = \{ s_i \} = (x_i, y_i) \) \((1 \leq i \leq M)\), label set: \( Y = \{ y_i \} \) \((1 \leq i \leq M)\), category set: \( C = \{ c_j \} \), number of selected features: \( N_1 \), number of weak classifier: \( T \), sample weight matrix: \( D = \{ D_{i,j} \} \) \((1 \leq i \leq T, 1 \leq j \leq M)\), threshold of noise confidence degree: \( th \).

**Output:** the final classifier \( H \).

**Step 1:** initialize the sample weight matrix:
1.1 for \( i = 1:1:M \)
1.2 \( D_{i,1} = 1/M \)
1.3 end for

**Step 2:** randomly sampling the training sample set \( TS \) to obtain the current training set \( ts_i \);
2.1 set the misclassified sample set: \( err_1 = \phi \).
2.2 for \( i = 1:1:T \)
2.3 \( ts_i = err_i \).
2.4 suppose the set of the samples which both belong to the category \( c_j(1 \leq j \leq L) \) and the misclassified sample set \( err_i \) is \( se_j \) then the training set \( ts_i \) is obtained using formula (22):
\[
\begin{align*}
    ts_i &= ts_i \cup \left( \bigcup_{j \in C} \text{RandSelect}\left(s_{cj} - se_j, nr_j\right) \right) \\
    nr_j &= m \times \frac{ne_j}{M} - ne_j
\end{align*}
\]  
(22)

where \( ne_j \) is the number of samples in \( se_j \), \( sc_j \) is the set of samples in category \( c_j \), the function \( \text{RandSelect}(sc_j - se_j, nr_j) \) randomly selects \( nr_j \) samples from the difference set of \( sc_j \) and \( se_j \), \( m \) is the maximum number of samples in \( ts_i \) \((m \text{ is set to } m = M/2 \text{ in this article})\) \( nc_j \) is the number of samples in \( sc_j \).

**Step 3:** calculate the noise confidence degree of the samples in \( ts_i \), obtain the noise sample set \( ns_i \) and update \( D_i \):
3.1 obtain the noise sample set (denoted as \( ns_i \)) in \( ts_i \) by using the proposed noise detection method (Algorithm 2).
3.2 for each \( s \in ns_i \)
3.2.1 \( D_{i+1,s} = 0 \)  
3.3 endfor

**Step 4:** select the best feature set \( fs_i \) from \( ts_i \) using Algorithm 3.

**Step 5:** train the SVM classifier on \( ts_i \) using \( fs_i \) to obtain the weak classifier \( h_i \).

**Step 6:** calculate the error rate \( \delta_i \) and classifier weight \( (\alpha_i) \) of \( h_i \) using formulas (21) and (20), then update \( D_i \):
6.1 predict the labels of the samples in \( ts_i \) using \( h_i \), and obtain the misclassified sample set \( err_{i+1} \).
6.2 calculate the corresponding error rate \( \delta_i \):
6.3 if \( \delta_i \geq (L - 1)/L \) or \( \delta_i = 0 \)
6.4 delete \( h_i \), continue.
6.5 end if
6.6 obtain the classifier weight \( \alpha_i \) using formula (20).

Algorithm (Continued) Improved Adaboost Algorithm Based on Noise Confidence Degree and Weighted Feature Selection (NW_Ada)

\[
D_{i+1,j} = \frac{D_{i+1,j}}{\sum_{s_i \in TS} D_{i+1,k}}
\]  
(25)

D. IMPLEMENTATION OF THE PROPOSED ALGORITHM

Compared to typical classifiers like Naive Bayesian (NB), KNN, Random Forest (RF), etc, the SVM classifier has the following advantages: (1) it can avoid overfitting problem by adjusting the parameters like penalty factor and kernel function. (2) it has low computational complexity in the label predicting process and it can avoid the dimensionality curse problem by using the support vectors. (3) it is robust and has high accuracy when dealing with the datasets with small sizes. On this basis, this article uses SVM as the weak classifier of Adaboost algorithm. Execution flow of the proposed algorithm is shown in Fig. 3 and the details are given in Algorithm 4:

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. EXPERIMENTAL SETUP

In this section, we use eight typical datasets which are downloaded from the UCI Machine Learning Repository [43, 44] for experiments. Table 5 gives the details including the sample number, feature number, category number, number of the maximum category, and number of the minimum category of each dataset. From Table 5 we know that the sample number of these datasets ranges from 300 to 4601, the feature number ranges from 34 to 10000, and the category number ranges from 2 to 13. Moreover, we know that most selected datasets are unbalanced and they are fit for testing the performances of different algorithms on dealing with the unbalanced datasets.

The experiment randomly selects 90% of the samples in each dataset as the training set and 10% of the samples as the testing set. For ease of comparisons, the number of iterations is set to \( T = 10 \), the number of neighbors in the noise detection algorithm is set to \( k = 10 \), and the threshold of
noise confidence degree is set to $th = 0.02$. For ensuring the classification accuracy, the parameters of SVM are set to: kernel function: RBF Kernel, penalty coefficient: $c = 200$, kernel function parameter: $g = 0.07$, where $c$ and $g$ are obtained by using Fly Optimization Algorithm (FOA) when $c$ ranges from 100 to 1000 with a step of 100 and $g$ ranges from 0.01 to 0.1 with a step of 0.01. Moreover, the experiment is carried out on the platform of 8G memory and Core i7 CPU, and all algorithms are implemented by using Matlab 2013. To measure the performances of different algorithms on dealing with unbalanced datasets, the macro-average $F_1$ measurement (called $F_{mac}$) which treats all categories equally is used in this article by combining the 10-cross validation method [39]. The definition of $F_{mac}$ is given in formula (26):

$$F_{mac} = \frac{1}{L} \sum_{i=1}^{L} F_i \quad (26)$$

where $L$ is the number of all categories, $F_i$ is the $F_1$ value on category $c_i$ which is obtained by using the $F_1$ measurement. The definition of $F_i$ is given as follows:

$$F_i = \frac{2 \times r_i \times p_i}{r_i + p_i}, \quad p_i = \frac{TP_i}{TP_i + FP_i}, \quad r_i = \frac{TP_i}{TP_i + FN_i} \quad (27)$$

where $p_i$ and $r_i$ are the precision and recall values on category $c_i$, $TP_i$ 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 belongs to category $c_i$ and are misclassified.

### B. COMPARISONS OF DIFFERENT NOISE DETECTION METHODS

When different datasets are used, three noise detection methods: the proposed noise detection method, the noise detection method in [27] and the noise detection method in [38] are applied to Step 3 of Algorithm 4, respectively. For ease of comparison, the other steps in Algorithm 4 remain unchanged and the related parameters are given in section V.A. As the ratio of selected features ($rf$) ranges from 0.1 to 0.5 with a step of 0.1, the average $F_{mac}$ values (denoted as $F_a$) obtained by Algorithm 4 which combines each of the above noise detection methods respectively are calculated and shown in Table 6.

### TABLE 6. Comparison of the $F_a$ values of different noise detection methods.

| datasets     | reference [38] | reference [27] | the proposed method |
|--------------|----------------|----------------|---------------------|
| Spambase     | 0.851          | 0.882          | 0.879               |
| AD           | 0.954          | 0.958          | 0.962               |
| KDD99        | 0.989          | 0.987          | 0.991               |
| DrivFace     | 0.973          | 0.985          | 0.989               |
| Arrhythmia   | 0.612          | 0.604          | 0.623               |
| AntiVirus    | 0.985          | 0.986          | 0.989               |
| Dermatology  | 0.945          | 0.956          | 0.972               |
| Amazon       | 0.781          | 0.789          | 0.783               |

According to Table 6 we know that, when comparing the proposed method with [38], the $F_a$ values of the proposed method are obviously higher in all datasets, which indicates that considering the deviated degree information of the samples can effectively reduce the noise detection error rate and improve the classification accuracy. When comparing the proposed method with [27], it is obvious that the corresponding $F_a$ values of the proposed method are higher except for the datasets of Spambase and Amazon. This may be due to the reason that the noise detection method in [27] uses SVM to obtain the labels of the sample neighbors, ignoring the fact that SVM classifier is difficult to deal with the unbalanced datasets.

Further, we compare the execution speeds of the above three noise detection methods. When $rf$ ranges from 0.1 to 0.5 with a step of 0.1, we obtain the average running time values (denoted as $rt$) of different methods, and the results are shown in Table 7. It can be seen that, as the proposed noise detection method considers the deviated degree information
of each sample, the corresponding \( r_t \) values are slightly higher than those of [38]'s method. When comparing with the method of [27], the proposed noise detection method has obvious advantage on execution speed especially when DrivFace dataset is used. That may be the reason that [27] uses the SVM classifier to predict the labels of the sample neighbors, which brings much more computation cost compared to the manner of using the real labels directly.

### C. COMPARISONS OF DIFFERENT FEATURE SELECTION METHODS

In order to verify the influence of sample weights on feature selection results, several traditional feature selection methods of filters (IG [40], CHI [41], IMGI [42] and CMFS [39]) are used for comparing with the corresponding weighted features selection methods (IGW, CHIW, IMGIW and CMFSW).

To make it fair, the above eight feature selection methods are applied to Step 4 of Algorithm 4, respectively, and the other steps in Algorithm 4 remain unchanged. Moreover, the related parameters are given in section V.A. On this basis, when the ratio of selected features (\( rf \)) ranges from 0.1 to 0.5 with a step of 0.1, the average \( F_{ac}\) values (denoted as \( F_{ac} \)) of each dataset when the ratio of selected features ranges from 0.1 to 0.5 with a step of 0.1. The results are shown in Fig. 5.

From Fig. 5, we know that, ND_Ada and Rob_Ada outperform Ada and SAMME when dealing with the datasets of Spambase, AD, and AntiVirus which have two categories. The performances of Rob_Ada and SAMME are generally better than those of ND_Ada and Ada when dealing with the datasets of Dermatology, KDD9, DrivFace, Arrhythmia, and Amazon which have more than two categories. Further, we notice that the methods combining \( r_{OR} \) can obtain higher \( F_{a} \) values than the methods using \( r_{OR} \) when dealing with the unbalanced datasets like Spambase, AD, KDD9 and DrivFace. For example, for the datasets of Spambase and DrivFace, the \( F_{a} \) values of ERR CR+Rob_MulAda are about 0.08 and 0.13 higher than those of ERR+Rob_MulAda, respectively, and the \( F_{a} \) values of ERR CR+SAMME is about 0.22 and 0.07 higher than those of ERR+SAMME, respectively. Therefore, we conclude that considering the category recall information in classifier error rate calculation process can effectively improve the classification accuracy of Adaboost algorithm on unbalanced datasets.

### D. COMPARISONS OF CLASSIFIER ERROR RATE CALCULATION METHODS

In this section, we denote the error rate obtained by the traditional error rate calculation method (ERR) as \( \epsilon_{j1} \), and denote the error rate obtained by the proposed category recall based error rate calculation method (ERR CR) as \( \epsilon_{j2} \). Table 9 gives the combinations of different error rate calculation methods and classifier weight calculation methods, where \( \epsilon_{j} \) is the error rate of classifier \( h_{j} \), \( \alpha_{j} \) is the classifier weight of \( h_{j} \), and \( L \) is the number of categories.

Like V.B and V.C, we also apply these combinations of different methods to Algorithm 4 respectively and keep the other steps unchanged. Then, we obtain the corresponding average \( F_{mac} \) values (denoted as \( F_{mac} \)) of each dataset when the ratio of selected features ranges from 0.1 to 0.5 with a step of 0.1. The results are shown in Fig. 5.

For the Spambase, AD, and AntiVirus which have two categories. The performances of Rob_Ada and SAMME are generally better than those of ND_Ada and Ada when dealing with the datasets of Dermatology, KDD9, DrivFace, Arrhythmia, and Amazon which have more than two categories. Further, we notice that the methods combining \( r_{OR} \) can obtain higher \( F_{a} \) values than the methods using \( r_{OR} \) when dealing with the unbalanced datasets like Spambase, AD, KDD9 and DrivFace. For example, for the datasets of Spambase and DrivFace, the \( F_{a} \) values of ERR CR+Rob_MulAda are about 0.08 and 0.13 higher than those of ERR+Rob_MulAda, respectively, and the \( F_{a} \) values of ERR CR+SAMME is about 0.22 and 0.07 higher than those of ERR+SAMME, respectively. Therefore, we conclude that considering the category recall information in classifier error rate calculation process can effectively improve the classification accuracy of Adaboost algorithm on unbalanced datasets.

### TABLE 7. Comparison of the \( r_t \) values of different noise detection methods (unit: second).

| datasets     | reference [38] | reference [27] | the proposed method |
|--------------|----------------|----------------|-------------------|
| Spambase     | 1.893          | 2.829          | 2.182             |
| AD           | 0.537          | 3.473          | 0.678             |
| KDD99        | 0.063          | 0.977          | 0.122             |
| DrivFace     | 0.056          | 0.513          | 0.079             |
| Arrhythmia   | 0.046          | 0.063          | 0.057             |
| AntiVirus    | 0.021          | 0.056          | 0.013             |
| Dermatology  | 0.015          | 0.026          | 0.019             |
| Amazon       | 0.032          | 0.086          | 0.038             |
E. COMPREHENSIVE COMPARISONS

In this section, we compare the proposed algorithm (NW_Ada) with six typical improved Adaboost algorithms (Ada_M1[13], SAMME[16], SAMME_R[17], AW_Ada[30], ND_Ada[26] and ROB_Ada[27]) when the number of iterations \(T\) ranges from 10 to 110 with a step of 20. The parameters of these algorithms are given as follows: 1) the number of neighbor samples in the noise detection process of ND_Ada, ROB_Ada and NW_Ada is set to \(k = 10\); 2) the feature selections in NW_Ada and the other algorithms are CMFSW and CMFS, respectively. As the ratio of selected features ranges from 0.1 to 0.5 with a step of 0.1, the average \(F_{ac}\) values (denoted as \(F_{ac}\)) are calculated and the results are shown in Fig. 6. It can be seen from Fig. 6 that, as the number
FIGURE 4. Comparison of $F_a$ values of different feature selections on different datasets.
FIGURE 5. Comparison of classification accuracy when different classifier error rate calculation methods and classifier weight calculation methods are used.
FIGURE 6. Comparison of $F_{ma}$ values of different algorithms under different iterations.
of iterations increases, though the performances of different methods fluctuate in some cases, the $F_a$ values of $T = 110$ are generally higher than those of $T = 10$. Moreover, we find that the performance of NW_Ada is significantly better than those of the other algorithms on the unbalanced datasets like KDD99, DrivFace and Dermatology. Further, we notice that although the performance of NW_Ada is similar to some of the other algorithms on the datasets like Spambase, AD, AntiVirus and Amazon in some cases, the $F_a$ values of NW_Ada are still generally higher than those of most algorithms, illustrating that our algorithm has significant advantage over the existing typical Adaboost algorithms on classification accuracy.

VI. CONCLUSION
In this article, an improved Adaboost algorithm for classification based on noise confidence degree and weighted feature selection is proposed. The main characteristics include: 1) To avoid the influence of noise samples, the conceptions of social degree and deviated degree are introduced; 2) As the existing feature selections of Adaboost algorithms ignore the differences of sample weights, a filter based weighted feature selection is proposed to improve the classification ability of dealing with misclassified samples; 3) In order to deal with the unbalanced datasets, a category recall based error rate calculation method is proposed to solve the problem that traditional error rate calculation methods always ignore the category distribution information of misclassified samples. In order to test the effectiveness of the proposed algorithm, experiments are carried out on the aspects of noise detection, feature selection, error rate calculation and comprehensive comparisons. Experimental results show that our algorithm outperforms the other algorithms on classification accuracy when different datasets especially the unbalanced datasets are used. In the future, we will study further from the following three aspects: 1) apply new sampling methods to improve the running speed of the proposed algorithm while guaranteeing the classification accuracy; 2) improve the diversity of weak classifiers to increase the classification accuracy and generalization ability of Adaboost based algorithms.

REFERENCES
[1] G. I. Webb and Z. Zheng. “Multistrategy ensemble learning: Reducing error by combining ensemble learning techniques,” IEEE Trans. Knowl. Data Eng., vol. 16, no. 8, pp. 980–991, Aug. 2004.
[2] L. Fan, “Hyperspectral image classification with limited labeled training samples using enhanced ensemble learning and conditional random fields,” IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens., vol. 8, no. 6, pp. 2427–2438, Jun. 2015.
[3] K. P. Singh, S. Gupta, and P. Rai, “Identifying pollution sources and predicting urban air quality using ensemble learning methods,” Atmos. Environ., vol. 80, pp. 426–437, Dec. 2013.
T. Hastie, S. Rosset, J. Zhu, and H. Zou, “Multi-class AdaBoost,”

X. W. Yang, Z. Ma, and S. Yuan, “Multi-class Adaboost algorithm based

T. F. Zhang, Q. Zhang, and J. Y. Liu, “URL classification method based

Y. Freund and R. E. Schapire, “Experiments with a new boosting

D. P. Solomatine and D. L. Shrestha, “AdaBoost RT: A boosting algo-

Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-

C. Zhuo, “Xgboost classifier for DDoS attack detection and analysis in

W. Lu, Z. Li, and J. Chu, “Adaptive ensemble undersampling-boost: A

B. Sun, S. Chen, J. Wang, and H. Chen, “A robust multi-class AdaBoost

R. A. Sowah, U. o. G. Computer Engineering Department, M. A. Agebure,

S. Chen, H. He, and E. A. Garcia, “RAMOBoost: Ranked minority

R. A. Sowah, U. o. G. Computer Engineering Department, M. A. Agebure,

M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera,

P. Thanh Noi and M. Kappas, “Comparison of random forest, k-nearest

Y. Yang and J. O. Pedersen, “A comparative study on feature selection in

J. Yang, L. Feng, and J. Zhu, “Novel artificial bee colony based fea-

G. Haixiang, L. Yijing, L. Yanan, L. Xiao, and L. Jinling, “BPSO-AdaBoost-KNN ensemble learning algorithm for multi-class imbalanced data classification,” Eng. Appl. Artif. Intell., vol. 49, pp. 176–193, Mar. 2016.

Y. Cao, “Improved behavior-based malware detection algorithm with AdaBoost,” J. Xidian Univ., vol. 6, pp. 116–124, Dec. 2013.

H. Xiao, Z. Xiao, and Y. Wang, “Ensemble classification based on super-

S. Yang, L. F. Chen, and T. Yan, “An ensemble classification algorithm for convolutional neural network based on AdaBoost,” in Proc. IEEE/ACIS 16th Int. Conf. Comput. Inf. Sci. (ICIS), 2017, pp. 401–406.

M. Yousefi, M. Yousefi, R. P. M. Ferreira, J. H. Kim, and F. S. Fogli-

Y. B. Chen, P. Dou, and X. J. Yang, “Improving land use/cover clas-

J. Yang, Y. Liu, X. Zha, Z. Liu, and X. Zhang, “A new feature selection

G. A. Mills, K. M. Koumadi, S. Y. Fiawoo, and U. f. D. S. Computer Sci-

S. Chen, H. He, and E. A. Garcia, “RAMOBoost:Ranked minority

H. Xiao, Z. Xiao, and Y. Wang, “Ensemble classification based on super-

P. Thanh Noi and M. Kappas, “Comparison of random forest, k-nearest

J. Yang, Y. Liu, X. Zha, Z. Liu, and X. Zhang, “A new feature selection

D. P. Solomatine and D. L. Shrestha, “AdaBoost RT: A boosting algo-

Y. Cao, “Improved behavior-based malware detection algorithm with AdaBoost,” J. Xidian Univ., vol. 6, pp. 116–124, Dec. 2013.

H. Xiao, Z. Xiao, and Y. Wang, “Ensemble classification based on super-

M. Yousefi, M. Yousefi, R. P. M. Ferreira, J. H. Kim, and F. S. Fogli-

Y. B. Chen, P. Dou, and X. J. Yang, “Improving land use/cover clas-

J. Yang, Y. Liu, X. Zha, Z. Liu, and X. Zhang, “A new feature selection

LIZHOU FENG was born in 1986. She received the Ph.D. degree from Jilin University, in 2015. She is currently a Lecturer with the Tianjin University of Finance and Economics. Her research inter-

YOUWEI WANG was born in 1987. He received the Ph.D. degree from Jilin University, in 2015. He is currently an Associate Professor with the Central University of Finance and Economics. His research interests include machine learning and data mining.

Youwei Wang

Lizhou Feng

IEEE Access