The Journal of Systems and Software 132 (2017) 272–282

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Adaptive Ensemble Undersampling-Boost: A novel learning framework for imbalanced data

Wei Lu, Zhe Li, Jinghui Chu*

School of Electronic Information Engineering, Tianjin University, Tianjin 300072, PR China

ARTICLE INFO

Article history:
Received 6 October 2016
Revised 3 March 2017
Accepted 6 July 2017
Available online 12 July 2017

PACS:
07.05.Kf

Keywords:
Classification
Imbalanced data sets
Real Adaboost
Voting algorithm
Adaptive decision boundary
Ensemble Undersampling

ABSTRACT

As one of the most challenging and attractive problems in the pattern recognition and machine intelligence field, imbalanced classification has received a large amount of research attention for many years. In binary classification tasks, one class usually tends to be underrepresented when it consists of far fewer patterns than the other class, which results in undesirable classification results, especially for the minority class. Several techniques, including resampling, boosting and cost-sensitive methods have been proposed to alleviate this problem. Recently, some ensemble methods that focus on combining individual techniques to obtain better performance have been observed to present better classification performance on the minority class. In this paper, we propose a novel ensemble framework called Adaptive Ensemble Undersampling-Boost for imbalanced learning. Our proposal combines the Ensemble of Undersampling (EUS) technique, Real Adaboost, cost-sensitive weight modification, and adaptive boundary decision strategy to build a hybrid algorithm. The superiority of our method over other state-of-the-art ensemble methods is demonstrated by experiments on 18 real world data sets with various data distributions and different imbalance ratios. Given the experimental results and further analysis, our proposal is proven to be a promising alternative that can be applied to various imbalanced classification domains.

1. Introduction

Classification is one of the most important tasks in machine learning. In the machine learning field, most classification algorithms are designed with the assumption that there is no significant difference between the number of samples in the majority class and that in the minority class. However, this prerequisite is always difficult to meet in real situations (Kang and Cho, 2006). Data imbalance occurs when the ratio between the number of samples in two classes is far from equal. In this situation, positive samples in the minority class are always under represented. As a result, the classification model tends to be dominated by negative samples in the majority class. In recent years, data imbalance has been widely regarded as a very challenging problem in data mining (Yang and Wu, 2006), especially for binary classification tasks, and it has appeared in a variety of classification tasks including breast cancer diagnosis (Parvin et al., 2013), credit card fraud detection (Dal Pozzolo et al., 2014), software defect prediction (Laradji et al., 2015), and protein-protein interactions (Zhang et al., 2012). For the above classification tasks, the samples in the minority class are likely to be abnormal and are, therefore, more important. Because the classic classifiers, such as Support Vector Machine (SVM) and Decision Trees, were designed without considering the data distribution, the minority class usually receives less attention.

Generally speaking, there are primarily two mainstream methods for tackling the data imbalanced problem, namely, sampling techniques and cost-sensitive learning. Sampling techniques operate on the data level and are widely used to provide a balanced distribution, among which oversampling and undersampling are the most representative methods. Oversampling methods aim to increase the number of positive patterns to balance the ratio between the two classes. The easiest way to produce oversampling is to copy the positive samples directly. This method will not lose any information, but the problem of overfitting might occur. For this reason, some advanced oversampling algorithms have been developed, such as SMOTE (Chawla et al., 2002), borderline-SMOTE (Han et al., 2005), ADASYN (He et al., 2008), and boundary oversampling (Li et al., 2016). However, these methods will still take a rather long time for training. Undersampling methods attempt to balance the ratio of different classes by reducing the number of negative patterns. Similarly, random undersampling (RUS), which means extracting samples randomly from negative patterns, is the easiest way to reach this goal. However, when the number of pos-

* Corresponding author.
E-mail addresses: luwei@tju.edu.cn (W. Lu), tywzlizhe29121@126.com (Z. Li), cjh@tju.edu.cn (J. Chu).

http://dx.doi.org/10.1016/j.jss.2017.07.006
0164-1212/

itive patterns is extremely low, the extracted samples might not represent the whole negative class and thus some important information will be lost. Ensemble of Undersampling (EUS) divides the whole negative set into several subsets, which are combined with the positive set (Kang and Cho, 2006). In this way, all of the negative patterns are used for training and the underrepresented problem is avoided. Similar to the oversampling techniques that choose to generate boundary positive samples, there also exists methods such as Bao et al. (2016) which does undersampling to only reserve borderline negative instances.Usually, undersampling techniques show better performance than oversampling techniques (Drummond et al., 2003). In addition to sampling techniques, there are many cost-sensitive algorithms that solve the imbalance problem at the algorithm level without changing the distribution of the whole data set. In contrast, they classify the patterns by finding the expectation that leads to the lowest cost (Elkan, 2001). Cost-based sampling (Barua et al., 2014) and weight modification strategies (Krawczyk et al., 2014) are representative traditional cost-sensitive algorithms. Cost-sensitive techniques can achieve good classification performance and control the losses due to misclassification at the same time.

In addition to the dominant techniques introduced above, there are also some other methods. Active learning, which is traditionally used on unlabeled training data, is now adopted for imbalanced classification (Ertekin et al., 2007). The one-class technique, which aims at building a classification model that has samples from only one specific class, is also applied in this field. One-class Support Vector Machines (SVMs) (Lee and Cho, 2006) and the auto encoder (Manevitz and Yousef, 2007) method are representative one-class techniques. Kernel-based algorithms are used to map the linearly nonseparable space into a higher dimensional space to help make the separation available. It can be combined with sampling methods (Tang and Zhang, 2006) as well as focused more on doing kernel modification (Hong et al., 2007). Moreover, some classic stochastic algorithms such as genetic programming (Mohd Pozi et al., 2015) and simulated annealing (Yousefian-Jazi et al., 2014) have also been widely applied in imbalanced classification tasks.

In recent years, ensembles and combination solutions (Fan et al., 1999; Chawla et al., 2003; Sun et al., 2007; Seiffert et al., 2010) have also become popular for solving imbalanced classification problems. Different strategies such as data distribution modification, cost-sensitive learning, bagging (Breiman, 1996), and boosting strategies can be combined and used simultaneously to improve the overall performance. According to Galar et al. (2012), the ensemble strategy performs better than the other non-ensemble methods. Multi-Boosting Webb (2000), SMOTEBoost (Chawla et al., 2003), RUSBoost (Seiffert et al., 2010), boundary-boost (Li et al., 2016), and BNU-SVM (Bao et al., 2016) are well-known boosting-based techniques. Bagging techniques, which are also known as bootstrap aggregating, are also used widely in the imbalanced classification field. EUS-SVM (Kang and Cho, 2006), SMOTEBagging (Wang and Yao, 2009), and EasyEnsemble (Liu, 2009) are representative algorithms among the Bagging algorithms. Boosting algorithms can also be combined with cost-sensitive techniques, such as Adacost (Fan et al., 1999). However, there are hardly any ensemble algorithms that focus on combining sampling, bagging and boosting together except for Multiboosting and EasyEnsemble, which are still short of providing good classification performance. Therefore, we believe that there is still potential to combine these three techniques and improve the voting and its threshold selection strategy.

Here, we propose a more effective ensemble learning framework. EUS is used for changing the data distribution and acquiring balanced subsets. For each individual classifier in EUS, we use Real Adaboost (Schapire and Singer, 1999) rather than the frequently used Discrete Adaboost to improve its accuracy. The results output by these classifiers are aggregated by a weighted voting system that is based on the error rate of each individual classifier. Finally, to make the final decision, we propose an adaptive threshold selection method that is based on the OTSU algorithm (Otsu, 1975) to find the optimal threshold that can bring the maximum between-class variance. The superiority of our proposal can be certified through empirical comparisons. Several metrics that are widely used to evaluate the performance of imbalanced classification techniques were adopted to make comparisons between our proposal and other classic and state-of-the-art methods. In the experiments, we chose C4.5 (Quinlan, 2014) as the base classifier for our experiments. We tested with 18 two-class imbalanced real-world data sets from the KEEL data set repository (Alcalá et al., 2010). The imbalance ratio of these data sets ranges from around 3 to 40. The main contributions of this paper can be summarized as follows:

1. We propose a novel framework for imbalanced classification tasks based on the combination of EUS and Real Adaboost.
2. We propose a weighting modification algorithm to give each individual classifier a weight for voting according to its own training error, which can help provide better performance, especially in highly imbalanced situations.
3. We introduce OTSU algorithm from the digital image processing field to determine the optimal threshold adaptively instead of a fixed voting threshold, thus obtaining better classification performance.

The remainder of this paper is organized as follows. In Section 2, we review the problems caused by imbalanced data. In Section 3, we propose our Adaptive Ensemble Undersampling-Boost (Adaptive EUSBoost) framework. Moreover, in this section, we also describe the data sets we use. Section 4 introduces the experimental settings and results, and then compares our proposal with some state-of-the-art techniques. Section 5 analyzes the reason why our proposal outperforms other methods. The conclusion is drawn in Section 6.

2. The data imbalanced problem

For binary classification tasks, a data set can be described as “imbalanced” if the number of patterns in the two classes is not the same. However, this problem is not very significant unless the gap is rather large. Because usually there are fewer positive instances than negative instances, the positive class tends to be underrepresented. As Fig. 1(b) shows, in this situation, the number of positive patterns is much smaller than the number of negative patterns and the original border can easily be invaded by the negative class. It is clear that if the classifier is trained in this way, it will be more likely to misclassify positive instances as negative instances. However, as we mentioned in Section 1, traditional classifiers always fail to overcome this problem because they were designed to find the decision border that can achieve high overall accuracy as well as good generalization performance.

Although the overall accuracy is traditionally used as the metric to evaluate the performance of the classifiers, it is not sufficient to use the overall accuracy alone in imbalanced classification tasks (He and Garcia, 2009). For example, if there exists a data set that has only 1 sample in the positive class and 99 samples in the negative class, the overall accuracy will be 99% if all of the samples are classified to the negative class (Kang and Cho, 2006). Obviously, this result counts for nothing even if an extremely high overall accuracy is achieved. From this point of view, it is necessary to introduce more effective metrics. On the other hand, the structure of the classifiers should also be improved. Because the space of the positive class is too sparse and easy to be invaded by the negative

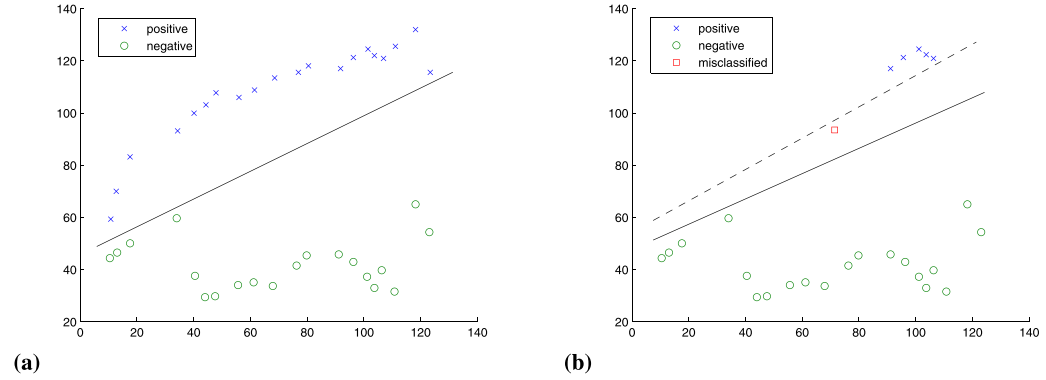

Fig. 1. The original decision boundary for two classes is shown in both a and b with a full line. One class is marked by crosses and the other class is marked by circles. When class imbalance occurs in b, the decision boundary is biased as according to the dotted line. The minority class is invaded and as a result, the pattern marked with a square is misclassified to the majority class. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

class, the exact border between the positive class and the negative class is difficult to find and positive patterns are likely to be wrongly classified to the negative class. As a result, classic classifiers cannot perform ideally when faced with imbalanced data sets. These problems will be discussed in the following sections.

3. Adaptive EUSBoost: ensemble undersampling guided boosting with an adaptive decision boundary

In this section, we introduce our proposal for alleviating the data imbalance problem. As mentioned above, our proposal is a hybrid framework that has ensembles of bagging, boosting and undersampling. In addition, we modify the weights of individual classifiers according to their classification error rate and introduce OTSU into our framework to find an optimal threshold instead of a predetermined fixed boundary for voting. By these methods, the imbalanced data problem can be alleviated effectively.

The organization of this section is as follows. In Section 3.1, we introduce the bias-variance decomposition, which is the theoretical basis of our proposal. In Section 3.2, we review the ensemble undersampling technique. Then in the next subsection, we combine the boosting algorithm with EUS and make a detailed description of how we design the adaptive determination boundary. Finally, the computational complexity of our proposal is discussed in Section 3.4.

3.1. Bias-variance decomposition: theoretical basis

For a supervised classification model, the bias-variance decomposition (Geman et al., 1992) is a method for analyzing a learning algorithm's generalization ability for a specific problem as a sum of three terms, namely, the bias, variance, and noise. Eq. (1) presents its mathematical formula:

$$error = bias^2 + variance + noise. \tag{1}$$

The bias is a term which is introduced by the choice of model and algorithm, and it quantifies the distance between the Bayes optimal function and the optimal function of the model itself. The variance is used to measure the inner deviation of the testing set which can be characterized as the deviation between the outputs and their average. The noise is the lower bound of the generalization error and measures the difficulty of the classification task itself. The generalization ability of a classifier can be improved by reducing either the bias or the variance. However, hardly any methods focus on reducing both the bias and variance to improve the overall performance. Here, we propose a novel method to achieve this goal.

3.2. Ensemble undersampling technique: variance reduction

Unlike random undersampling which uses only one subset of the majority class, EUS (Kang and Cho, 2006) makes use of all of the negative patterns to ensure the diversity and representativeness of the patterns. First, the imbalance ratio (*IR*) which indicates the skewness of a binary-class data set is computed. The definition of *IR* is shown in Eq. (2):

$$IR = \frac{n(negative)}{n(positive)}, \tag{2}$$

where *n(negative)* and *n(positive)* represent the number of negative and positive patterns respectively. Then, the majority class patterns are randomly undersampled without repetition *IR* times to construct *IR* negative subsets. Afterward, the positive set, which contains all of the minority patterns, will be combined with each negative subset, respectively, to build *IR* perfectly balanced training sets before they are used to train a certain classifier. In contrast to traditional bagging-based algorithms, the samples used for training are extracted without replacement. Finally, the outputs are aggregated by a predetermined rule such as majority voting, weighted voting or function value aggregation. The structure of EUS is shown in Fig. 2.

As observed from Fig. 2, all of the negative patterns are used for training. In this way, EUS overcomes the problem of underrepresentation caused by random undersampling and ensure the data diversity to help increase the generalization ability. On the other hand, compared with oversampling techniques, EUS reduces the computational complexity due to having fewer patterns used in the training of each individual classifier. In addition, because the EUS technique consists of *IR* balanced independent and identically distributed individual classifiers, the total error variance can be reduced by a factor of *IR*. Consequently, EUS results in good classification performance and efficiency (Kang and Cho, 2006).

3.3. Combining EUS with boosting and an adaptive boundary decision strategy

EUS can achieve good performance in imbalanced classification tasks because it can reduce the classification variance, ensure the diversity of the data, and maintain rather low computational complexity. However, the performance can easily be influenced by the classification ability of the base classifier. When the bias of the base classifiers is rather high, the classifier tends to misclassify a large number of patterns, and EUS cannot achieve good performance. For this reason, to obtain a better framework, it is important to reduce the bias and promote the performance of each base classifier.

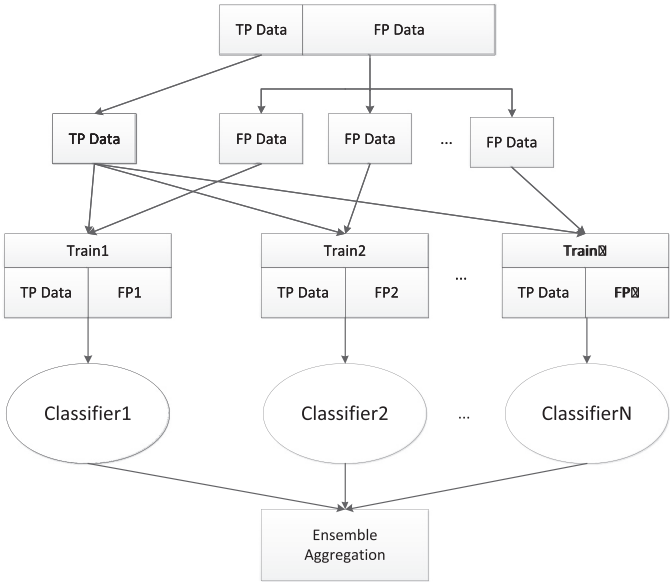

Fig. 2. The structure of EUS. The individual classifiers are trained by different subsets individually. Generally speaking, the number of subsets is approximately equal to the imbalance ratio.

Boosting (Freund and Schapire, 1997), which is quite effective and easy to implement, is widely used to integrate individual classifiers to promote performance. In this way, the bias of the base classifiers can be reduced. Adaboost.M2 (Freund and Schapire, 1997) is the most popular boosting strategy that is used for imbalanced classification (Liu, 2009; Seiffert et al., 2010; Galar et al., 2013). When used in conjunction with many other types of learning algorithms, Adaboost.M2 adjusts the weight of subsequent weak learners in favor of the instances that were misclassified by previous classifiers.

However, the output of each weak learner in Adaboost.M2 is either -1 or $+1$, which is insufficient. Therefore, we use Real Adaboost(Schapire and Singer, 1999), which is confidence-rated, to obtain a more accurate output. The output of each classifier is a real number, which can be viewed as the confidence level. Real Adaboost uses a fuzzy set to take the place of classical set so as to improve the performance of the whole classifier. The real outputs are aggregated and compared with a preset threshold to obtain the final classification result.

In our proposal, Real Adaboost is combined with each individual classifier in the EUS framework to achieve better performance. All of the individual outputs are aggregated and a final output that will be generated for each pattern in the testing set. Voting is needed to fulfill this task. According to Kang and Cho (2006), classic voting strategies, such as majority voting, weighted voting, and functional value aggregation, show no significant difference. However, all of these techniques have a fixed predetermined decision boundary instead of a non-adaptive boundary. Moreover, the weight modification strategy, which can give every individual classifier a different weight according to its error rate, can also be improved. Our proposal is illustrated in Algorithm 1.

In Algorithm 1, each sample is represented as (x_i, y_i). Here, x_i is the feature vector of one sample, and y_i, which can be either $+1$ (positive) or -1 (negative), is its class label. Assume that the training set consists of m samples that are extracted from the whole data set with the original imbalance ratio. These instances are undersampled by EUS and thus, we can obtain IR balanced individual classifiers. Then the remaining $(n\text{-}m)$ patterns which are also with the original IR are used as the validation set. Let each base classifier that is trained by Real Adaboost be h_i. These patterns are classified by the EUSBoost framework, and the original weight of each individual classifier α_i will be calculated logarithmically based on the error rate ϵ_i. Finally, the weight W_i is calculated by normalizing the α_i. The reason that we use the log-loss measurement is that it incorporates the idea of probabilistic confidence and, thus, introduces the concept of cross entropy. Cross entropy incorporates the entropy of the true distribution into extra unpredictability when one assumes a different distribution from the true distribution. As a result, log-loss is an information-theoretic measure to gauge the "extra noise" that comes from using a predictor opposed to the true labels. By minimizing the cross entropy, the accuracy of the classifier can be maximized. As a result, it is estimated that the log-based weight can achieve better performance than other widely-used functions, such as the inverse proportion loss. We will also make comparison in Section 4 to verify its correctness.

To aggregate the output from each individual classifier, a threshold is always predetermined. For binary classification tasks that have negative patterns labeled as -1 while positive patterns are $+1$, the threshold is usually set to 0. However, 0 might not be the best threshold due to the different data distributions. Our proposal introduces OTSU (Otsu, 1975) from the image processing field, which aims at obtaining the optimal threshold by finding the maximum between-class variance, to obtain an adaptive threshold. Each pattern in the validation set r_k will be aggregated by log-based weighted voting, and the final result will be a real number. Then OTSU will find the optimal threshold for all of the results. In this way, the threshold is non-fixed and can change according to the data distribution, which helps the whole Adaptive EUSBoost framework achieve better testing accuracy. Because the threshold th ranges from -1 to 1, we set the step to be 0.01 and find the optimal threshold th_{opt} that can get the maximum between-class

Algorithm 1 Adaptive EUSBoost.

Input: Training set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$; Validation set $\{(x_{m+1}, y_{m+1}), (x_{m+2}, y_{m+2}), \ldots, (x_n, y_n)\}$; Imbalance Ratio *(IR)*

Output: $H(x) = sign\left(H(x) = \sum_{i=1}^{IR} W_i h_i(x) - th_{opt}\right)$

1: **for** i=1 to IR (IR is the imbalanced ratio) **do**
2: Build the majority subset by random sampling without replacement from the majority class whose size is equal to that of minority class;
3: Construct the training subset by combining the majority subset and minority class;
4: Train the individual classifier with Real Adaboost as h_i;
5: $\varepsilon_i = \sum_{j=m+1}^{n} \frac{1}{n-m}\left[y_j \neq h_i(x_j)\right]$, here, ε_i describes the normalized error rate of the classifier h_i on the validation set;
6: $\alpha_i = \frac{1}{2}\ln\frac{1-\varepsilon_i}{\varepsilon_i}$, here, α_i is the weight of classifier h_i;
7: $W_i = \frac{\alpha_i}{Z}$, here, W_i is the weight of each individual classifier, Z is for normalization, whose value is the sum of α_1 to α_{IR};
8: **end for**
9: // OTSU decision boundary search (validation)
10: **for** k=m+1 to n **do**
11: $$r_k = \sum_{i=1}^{IR} W_i h_i(x_k)$$
12: $$th_{opt} = \arg\max_{th} \sigma^2\left\{\overline{r_k \leq th}, \overline{r_k \geq th}\right\},$$
13: where th_{opt} is the optimal adaptive decision boundary.
14: **end for**
15: Output final classifier:
$$H(x) = sign\left(\sum_{i=1}^{IR} W_i h_i(x) - th_{opt}\right)$$

variance th_{opt}. In Algorithm 1, let r_k be the output of one validation sample, and then the validation samples can be divided into two parts, namely, samples that obtain an r_k value less than th and samples that have an r_k value no less than th. These two classes have their own centers, $\overline{r_k \leq th}$ and $\overline{r_k \geq th}$, respectively, and the threshold that can obtain these two centers is just the optimal threshold th_{opt}.

3.4. Computational complexity

The computational complexity of our proposal arises mainly from two sources: the training of the Real Adaboost model and the result aggregation.

In Real Adaboost model training, because the individual classifiers are trained simultaneously, the computational complexity will be similar to that of RUSBoost, which is at $O(nTM)$, where n,T,M stands for the number of samples in each training set, the number of iterations that Real Adaboost is executed and the number of features respectively. In our proposal, because all of the positive patterns are used for training in each balanced individual training set, n is approximately twice the size of the positive samples. Therefore, over-sampling techniques such as SMOTEBoost will be computationally more expensive than our proposal as a result of having a larger training set. However, compared with RUSBoost, we acknowledge that our proposal is still slightly more computationally consuming. For the result aggregation, because an OTSU-based algorithm is used to find the optimal decision boundary, there is an additional complexity of $O(n)$. Therefore, the overall computational complexity of our proposal is $O(nTM+n)$. Thus, the complexity of the framework is a little higher than but still close to $O(n)$.

In general, because our proposal is based on the undersampling technique, the computational complexity will decrease with the increase in IR. Moreover, for one data set, the training time will be considered, once as long as there is no need for on-line classification. In conclusion, although our proposal is not the computationally cheapest ensemble method, it is still close to the best.

4. Experimental framework

This section presents our experimental design. In Section 4.1, we give a detailed description to the data sets used in our experiments. Then, in Section 4.2, we illustrate the base learning algorithm that we used. Finally, for the methods used for comparison, the related parameters are set in Section 4.3. Metrics that are used for evaluating the performance of algorithms are introduced in Section 4.4. The framework used for comparison between our proposal and other baselines and state-of-the-art methods is introduced in Section 5.

4.1. Data sets

All of our data sets are from the KEEL data set repository (Alcalá et al., 2010). The repository is available on the corresponding webpage[1]. We consider 18 imbalanced binary data sets from various application domains. Table 1 presents a brief description of these data sets, including the name, number of patterns, number of original multi-class attributes (#Attr), number of positive patterns (#Pos), proportion of positive patterns in the whole data set (%Pos) and IR of each data set. The imbalance ratio ranges from approximately 3 to 40.

[1] http://www.keel.es/dataset.php.

Table 1
Data set characteristics.

ID	Data sets	Size	#Attr	#Pos	%Pos	IR
1	Vehicle3	846	18	212	25.06	2.99
2	New-thyroid1	215	5	35	16.29	5.14
3	Segment0	2308	19	329	14.25	6.02
4	Yeast3	1484	8	163	10.99	8.1
5	Yeast2vs4	514	8	51	9.92	9.08
6	Ecoli067vs5	220	6	20	9.09	10
7	Ecoli01vs5	240	11	20	8.33	11
8	Ecoli0147vs56	332	6	25	7.53	12.28
9	Yeast1vs7	459	7	30	6.54	14.3
10	Glass4	214	9	13	6.07	15.47
11	Ecoli4	336	7	20	5.95	15.8
12	Abalone9-18	731	8	43	5.75	16.4
13	Yeast1458vs7	693	8	30	4.33	22.1
14	Flare-F	1066	11	43	4.03	23.79
15	Car-vgood	1728	6	65	3.76	25.58
16	Winequality-red4	1599	11	53	3.31	29.17
17	Kr-vs-k3vs11	2935	6	81	2.76	35.23
18	Abalone17vs7to10	2338	8	58	2.48	39.31

4.2. C4.5: the base weak learner

To make an empirical comparison, all of the ensemble methods included in our experimental framework should have the same base classifier. Moreover, the parameters of this base classifier should be identical. As one of the top 10 algorithms in data mining (Wu et al., 2008), C4.5 (Quinlan, 1993) is a type of decision tree that can generate easily understood classification rules as well as achieve ideal accuracy. It has also been applied widely for imbalanced classification techniques (Seiffert et al., 2010; Galar et al., 2013; Fernández et al., 2015). In this paper, C4.5 was implemented as J48 in Weka Hall et al. (2009). We used the non-pruning tree and set the confidence level to 0.25. The number of patterns per leaf should be no less than 2. All of the algorithms for comparison use the same configuration.

4.3. Experimental setup

As mentioned in Section 1, ensemble methods usually outperform other methods in this field. Therefore, to make an empirical comparison, we select some state-of-the-art ensemble methods, many of which were also considered in Galar et al. (2013). Finally, as mentioned in Section 3.3, in our proposal, both the log-based weight modification strategy and the inverse proportion-based strategy could work. It is expected that the log-based strategy performs better. To prove that, our two strategies (the log-based one is annotated as AEB_L, and the inverse proportion-based one is annotated as AEB_I) were used in comparisons with other methods. The testing results can be observed in Section 5. The algorithms that were used for the empirical comparison are listed in Table 2.

For SMOTE and Evolutionary Undersampling, there are some parameters that must to be set. In SMOTE, the number of neighborhood candidates for each positive pattern was set to be 4. The SMOTE process will stop when a balanced data set is obtained. For Evolutionary Undersampling, these configuration parameters were set to the same values as in Galar et al. (2013). All of the algorithms are implemented in Matlab 2014a with a Weka interface to implement C4.5. Because there are not many positive patterns in some of the data sets, to ensure that the test partition has enough patterns to be representative, we tested all of the algorithms listed in Table 2 with five-fold cross-validation instead of with ten-fold cross-validation. For our proposal, because we extracted a validation set to obtain the optimal decision boundary and individual weight, cross-validation was not available. However, to make the

Table 2
State-of-the-art ensemble algorithms for comparison.

Abbr.	Algorithm	Comments
SBA	SMOTEBagging (Wang and Yao, 2009)	Use SMOTE to generate positive pattens in bagging.
UBA	Underbagging (Barandela et al., 2003)	Use RUS to extract negative patterns in bagging.
EUS	Ensemble Undersampling (Kang and Cho, 2006)	Use RUS to build several balanced subsets.
SBO	SMOTEBoost (Chawla et al., 2003)	Adaboost.M2 is combined with SMOTE.
RBO	RUSBoost (Seiffert et al., 2010)	Adaboost.M2 is combined with RUS.
EAE	EasyEnsemble (Liu, 2009)	Each bag is built by RUS and then trained by Adaboost.M2.
EVU	Evolutionary Undersampled Boosting (Galar et al., 2013)	Adaboost.M2 is combined with evolutionary undersampling.
BBO	Boundary-Boost (Li et al., 2016)	Oversample boundary samples and train the classifier using Adaboost.M2.
BNU	Boosted Near-Miss Undersampling (Bao et al., 2016)	Undersample to keep the borderline samples for building boosted training sets.

Table 3
Confusion matrix in binary imbalanced data sets.

	Positive Prediction	Negative Prediction
Positive instance	TP	FP
Negative instance	FN	TN

experimental environment similar, we try to simulate the five-fold cross-validation process. We used half of the whole data set as the training set. The remaining patterns in the data set were divided equally into two parts. One was used as the validation set, and the other was used for testing. The division of training, validation, and testing sets is random. This process was repeated 5 times.

4.4. Metrics for performance evaluation

Traditionally, the overall accuracy is observed as the most important metric to evaluate the performance of a classifier. For binary classification tasks, the overall accuracy can be calculated by Eq. (3), as follows:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}, \tag{3}$$

where *Acc* stands for the overall accuracy, and *TP, TN, FP, FN* are the abbreviations of the true positive, true negative, false positive, and false negative, respectively, as shown in the confusion matrix in Table 3.

However, for imbalanced classification tasks, people usually focus more on the accuracy of each class and the overall accuracy is not suitable for analyzing imbalanced data sets. Nevertheless, because the positive instances are far less than the negative samples in imbalanced classification tasks, *TN* plays a more important role than *TP* in the overall accuracy. As a result, classifiers will classify more samples as negative to achieve a high overall accuracy. To evaluate the performance of the classifiers more precisely, some alternative metrics are proposed. More alternative metrics, including *sensitivity* (*recall*), *specificity*, and *precision* that can evaluate the performance more effectively can be deduced by the confusion matrix in Table 3.

The *precision* and *recall* are often employed together in tasks that regard one class as being more important than the other. However, one single metric is still insufficient because a better recall sometimes means that the precision is not good enough. Metrics that measure the trade-off between the recall and precision can reflect the performance of a classifier better. $F_1 - measure$ and $G - mean$ are metrics that are suitable for this purpose.

Despite the numerical metrics mentioned above, the performance of a classifier can be evaluated by the Receiver Operating Characteristic (ROC) (Fawcett, 2006) graph. The ROC can visualize the trade-off between the TP and FP rate (Galar et al., 2013). The Area Under the Curve (AUC) (Lobo et al., 2008) is a numeric metric that corresponds to ROC, and it indicates the area of the region under the ROC curve. Better classifiers often have higher AUC values because these classifiers can achieve high sensitivity and specificity at the same time. The AUC has been applied in various imbalanced classification tasks (Galar et al., 2013; Menardi and Torelli, 2014; Zhang et al., 2015). Eq. (4) presents the formulation to calculate the AUC:

$$AUC = \frac{1 + TP_{rate} - FP_{rate}}{2}. \tag{4}$$

5. Experimental results and empirical comparisons

In this section, several empirical comparisons are made to illustrate that our proposal outperforms other state-of-the-art ensemble techniques in imbalanced classification tasks. In addition, we also present that a log-based weight modification strategy achieves better performance than the inverse proportional strategy.

Among the performance metrics mentioned in Section 4.4, the AUC, which considers both the TP rate and FP rate, can provide a single scalar value to assess the performance of a classifier (Fawcett, 2004). Therefore, AUC is usually one of the most considered performance metrics for the performance evaluation of imbalanced classifications. In Table 4, we present the AUC values of all of the algorithms for each data set.

In addition to the AUC simulation results of the algorithms on each data set, the average AUC value for each algorithm is also presented in Table 4, followed by the winning times and the times that an algorithm ranks in the top 3 of 18 data sets. Based on the results shown in Table 4, we can find that our proposals show better performance than other state-of-the-art ensemble methods. Furthermore, the log-based Adaptive EUSBoost performs slightly better than the inverse proportion-based method, which verifies our anticipated result in Section 3.2. To demonstrate the superiority of our proposal better, some statistical methods including box plot, Friedman test, and Nemenyi post-hoc test are also introduced, as follows.

Box plot is a method of summarizing a set of data are measured on an interval scale which is often used in exploratory data analysis. The AUC box plot of all of the algorithms for comparison in this paper is shown in Fig. 3.

The box plot can give a clear and qualitative description of the performance of classifiers. However, it has less quantitative analysis and cannot ensure whether the superiority of our proposal is by accident. The Friedman test, which is also a non-parametric statistical test, is a good method for solving this problem. It can be used to detect differences across multiple algorithms. The procedure involves ranking algorithms on each algorithm together and then considering the values of the ranks by the data sets. We assume that the performances of all of the techniques for comparison are the same and set the *p*-value at 0.05.

According to our experimental result, the Friedman test has rejected the hypothesis that all of the algorithms in the comparison have the same performance, with an extremely low *p*-value ($p < 0.0001$), which means that the performance of the techniques used in the comparison show significant difference from one an-

Table 4
Performance evaluation with AUC. The best result in each data set is in bold and the top 3 results in each data set are underlined.

ID	SBA	UBA	EUS	SBO	RBO	EAE	EVU	BBO	BNU	AEB_L	AEB_I
1	0.7488	**0.7910**	0.7442	0.7455	0.7634	0.7282	0.7713	0.7452	0.7605	0.7836	0.7614
2	0.9719	0.9696	0.9607	0.9623	0.9623	0.9481	0.9576	**1.0000**	0.9597	**1.0000**	0.9861
3	0.9908	0.9853	0.9853	**0.9957**	0.9918	0.9873	0.9914	0.9924	0.9891	**0.9957**	0.9949
4	**0.9331**	0.9134	0.9141	0.8772	0.9039	0.9249	0.9245	0.9007	0.8945	0.9313	0.9294
5	0.8800	0.9415	0.9249	0.9114	0.8941	0.9045	0.9280	0.9152	**0.9617**	**0.9479**	0.9431
6	**0.8975**	0.8700	0.8175	0.8875	0.8775	0.8750	0.8875	0.8750	0.8625	0.8875	0.8750
7	0.8523	0.8545	0.8772	0.8841	0.9091	0.8545	0.8523	0.8847	0.9298	0.9136	**0.9341**
8	0.8922	0.8600	0.8531	0.8923	0.8547	0.8719	0.8992	0.8663	0.8427	**0.9054**	0.8887
9	0.7315	0.7572	0.7614	0.7616	0.7041	0.7105	0.6904	0.7617	0.7084	**0.7775**	0.7710
10	0.8742	0.8347	0.8259	0.8292	0.9005	0.8830	0.8650	0.8536	0.9001	**0.9097**	0.8876
11	0.9049	0.8617	0.8751	0.9342	0.9180	0.9180	0.8901	**0.9529**	0.9513	0.9433	0.9449
12	0.7368	0.7791	0.7597	0.7550	0.7591	0.7280	0.7312	0.7670	0.7343	**0.7803**	0.7775
13	0.5571	0.6113	0.5612	0.6055	0.6134	0.6017	0.5597	0.6233	0.6179	**0.6599**	0.6568
14	0.7792	0.8316	0.8262	0.7214	0.8066	0.8304	0.7619	0.8147	0.7672	**0.8459**	0.8344
15	0.9781	0.9678	0.9678	0.9766	0.9720	0.9586	0.9594	0.9454	0.9387	**0.9819**	0.9780
16	0.6610	0.6408	0.6347	0.6211	0.5792	0.6080	0.6551	0.6261	**0.6809**	**0.6791**	0.6640
17	0.9915	0.9804	0.9904	**0.9938**	0.9904	0.9702	0.9918	0.9826	0.9850	0.9877	0.9730
18	0.7949	0.8060	0.8304	0.7532	0.8007	0.7919	0.8039	0.8113	0.8265	**0.8540**	0.8459
Average	0.8445	0.8475	0.8394	0.8393	0.8445	0.8386	0.8400	0.8510	0.8506	**0.8769**	0.8691
Winner	2	1	0	2	0	0	0	2	2	12	1
Top 3	4	3	1	4	1	0	4	4	5	16	13

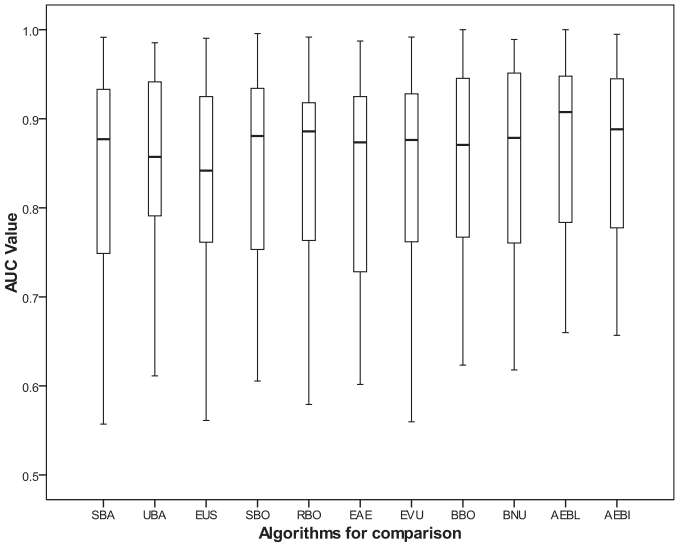

Fig. 3. The AUC box plot. In this figure, box plot describes the highest, lowest, upper and lower quartiles, and the median AUC values of each algorithm. Generally speaking, good algorithms have a high average AUC as well as a short box length.

Homogeneous Subsets

		Subset		
		1	2	3
Sample	AEBL	1.944		
	AEBI		3.556	
	BBO		5.861	5.861
	SBA		6.194	6.194
	SBO		6.444	6.444
	BNU		6.444	6.444
	RBO		6.472	6.472
	UBA		6.528	6.528
	EVU		6.861	6.861
	EUS			7.472
	EAE			8.222
Test Statistic		0.2	13.991	7.152
Sig. (2-sided test)		.	0.051	0.52
Adjusted Sig. (2-sided test)		.	0.07	0.593

Fig. 4. The homogeneous subsets of all of the algorithms used in the comparison. The subsets are generated based on asymptotic significances. The significance level is set at 0.05. A total of 11 algorithms are divided into three subsets. The algorithms in one specific subset show no significant classification performance.

other. In addition to this conclusion, all of the algorithms can also be divided into several homogeneous subsets, each of which contains algorithms that have similar performances. Fig. 4 presents these subsets.

To further differentiate these algorithms as well as demonstrate the correctness of the conclusions drawn from the Friedman test, the Nemenyi post-hoc test is also adopted in this paper. The Nemenyi test is intended to find the groups of data that differ after a statistical test of multiple comparisons (such as the Friedman test) has rejected the hypothesis that the performance of the comparisons on the groups of data is similar. The test makes pair-wise tests of the performances. It computes an average ranking difference threshold *CD*, and the hypothesis "the performance of two algorithms is the same" will be rejected if their average ranking difference is smaller than *CD*/2. Fig. 5 shows the test image.

Several conclusions can be drawn from the figures above:

1. Fig. 3 demonstrates that our two proposals, especially for the log-based method (AEB_L), outperform other algorithms with a rather significant advantage. For the AEB_L, the median is the highest and the distance between the 25th and 75th percentiles is rather small, which indicates that the overall performance is quite good and steady. The quantitative analysis from Fig. 4 also draws a similar conclusion. AEB_L shows the best ranking performance and it is the only element of the first subset. In contrast, EAE and EUS show the worst performance and they are only contained in the third subset (in blue). The remaining 9 algorithms are composed of two partly overlapping subsets. BBA, SBA, SBO, BNU, UBA, RBO, and EVU appear in both the second and the third subsets. However, AEB_I performs better than them and appears in only the second subset, which means that it still outperforms the other methods even though its performance is not as good as AEB_L.

2. Fig. 5 shows that there is no significant difference between AEB_L and AEB_I, however, other methods perform significantly less

Table 5
Performance evaluation with Specificity, Sensitivity (Recall), Precision, $F_1 - measure$ and Geometric Mean (G-mean). The best result in each metric in boldface.

Performance Metric	SBA	UBA	EUS	SBO	RBO	EAE	EVU	BBO	BNU	AEB_L	AEB_I
Specificity	0.9152	0.8433	0.7692	**0.9418**	0.8799	0.8367	0.8622	0.8803	0.8441	0.8513	0.8479
Sensitivity	0.7733	0.8531	**0.9098**	0.7394	0.8103	0.8424	0.8215	0.8295	0.8577	0.9006	0.8906
Precision	0.4690	0.3678	0.3352	**0.5536**	0.4149	0.3716	0.4131	0.4470	0.4084	0.4742	0.4099
$F_1 - measure$	0.5664	0.4826	0.4464	**0.6211**	0.5214	0.4822	0.5162	0.5349	0.5163	0.5405	0.5185
G-mean	0.8339	0.8475	0.8313	0.8256	0.8405	0.8389	0.8398	0.8498	0.8516	**0.8737**	0.8657

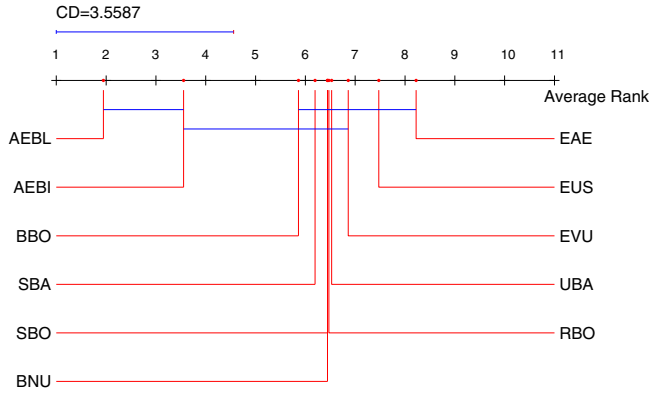

Fig. 5. The Nemenyi test image. $CD = 3.5587$. When the rank gap between two different algorithms exceeds $CD/2$ (i.e. around 1.8), the performance of these two algorithms can be viewed as significantly different.

well because the average ranking difference exceeds 1.8. As a result, our proposals outperform other state-of-the-art methods. This conclusion is similar to that drawn from the Friedman test.

Table 5 presents the results of evaluating each ensemble technique using the other performance metrics mentioned in Section 4.4. For each metric, the table shows the average value of every algorithm on all of the 18 data sets that we used.

It can be observed that our two proposals rank top 3 in more than 2/3 data sets regarding AUC, and AEB_L wins 12 out of 18 times. Therefore, we can claim that our proposals show the best AUC performance among all algorithms for comparison. It can also be verified by Table 5 that our two proposals show a good trade-off between the sensitivity and specificity. As a result, they always win when considering the AUC metric, and the two proposed methods are usually very close to the best algorithms by means of specificity and $F_1 - measure$. We can see from Table 5 that SMOTEBoost leads in specificity, precision, and $F_1 - measure$. This finding arises because the oversampling techniques copy or generate a large number of positive samples that are quite similar to the existing samples, which causes an overfitting problem. Furthermore, because our testing sets maintain the original *IR*, testing sets will mainly constitute of negative samples. Thus, the number of misclassified negative samples will also be greater, which will lead to a degradation in precision and $F_1 - measure$. In this situation, sensitivity and G-mean are more proper evaluation metrics. Finally, positive samples still tend to be misclassified in SMOTE and SMOTEBoost. They actually sacrifice the sensitivity to gain higher specificity, precision, and $F_1 - measure$. The advantage of oversampling techniques in specific metrics in Table 5 cannot lead to the conclusion that our proposals are not outstanding. For imbalanced classification tasks, the sensitivity is more important than the other metrics.

Except for the conclusions drawn above, the results presented in Tables 4 and 5 also reveal the following facts.

First, the sensitivity values for the undersampling methods are generally higher than those of the oversampling methods. The reason is that oversampling can cause the problem of overfitting because some noisy data can be replicated or generated many times. Moreover, the high computational expense is also a major drawback. In contrast, the influence of noisy data can be significantly reduced in undersampling methods because most of them are excluded from the negative subset. For most imbalanced data sets that have enough positive samples, the underrepresentation problem of the negative subset can be avoided effectively. As a result, compared with oversampling-based techniques, undersampling-based techniques have better performance in these data sets. However, in data sets that have fewer positive samples, such as Glass4 and Ecoli4, the information on the negative set is lost seriously and therefore the model cannot reflect the data distribution accurately. In this situation, undersampling techniques are not as competitive as oversampling techniques.

Second, EUSBoost, which is a key component of our proposal, improves the performance of each individual classifier by combining Real Adaboost with the classification algorithm. Moreover, the EUS algorithm in EUSBoost provides good data diversity as well as reducing the variance of a classifier, which can ensure the robustness of our proposal. Compared with RUS, EUS uses all of the instances in the original data set. By doing so, the diversity of the data remains, and the problem of underrepresentation is solved. Furthermore, the variance of the whole proposal is reduced by a factor of approximately *IR* compared with SMOTE because the *IR* subsets are used for training individually. These advantages can help our proposal obtain better classification results.

Third, the voting system with the weight modification strategy and the adaptive decision boundary also contribute to better performance. Our weight modification strategy is cost-sensitive, and the weight that is attributed to each individual learner is related to its classification error rate. Individual classifiers that can provide better accuracy on the validation set will get greater weights. In this way, the performance can be promoted. In addition, we can see from Table 5 that our two proposals show fairly good results in terms of both sensitivity and specificity. At the same time, the gaps between the sensitivity and specificity in our proposals are rather trivial. As a result, our proposals also have high geometric mean values, which means that our proposals can alleviate the imbalanced problem effectively while keeping a high sensitivity to the negative patterns. The reason is that our adaptive decision boundary method will find the threshold that can ensure the largest between-class variance.

Overall, the evaluation results provided in Tables 4 and 5 illustrate that our proposal behaves excellently on all major performance metrics, especially for those metrics that can reflect the trade-off between the positive and negative classification performance (AUC, $F_1 - measure$ and G-mean). Moreover, as anticipated, the log-based weight modification strategy performs better than the inverse proportional based strategy. However, when there are not enough positive patterns in the original data set, the negative patterns in each subset might not be able to represent the whole class effectively. This problem decreases the performance of Adaptive EUSBoost to some extent.

6. Concluding remarks

As a challenging and active research hotspot in machine learning, data mining and pattern recognition, imbalanced learning always receives sufficient attention in both theoretical and practical fields. In this paper, we propose a novel hybrid ensemble algorithm called Adaptive EUSBoost to alleviate the negative influence that is caused by imbalanced data sets. On several data sets with different imbalance ratios, we tested our proposal and compared it with some state-of-the-art ensemble methods. Experimental results prove that Adaptive EUSBoost shows favorable performance in the empirical comparison.

Adaptive EUSBoost is an effective and efficient ensemble imbalanced learning framework. It remains a relatively low computational expense as well as ensuring ideal data diversity. Moreover, the application of the log-based weight modification strategy and the adaptive decision boundary also promote the classification performance. Our experiments were conducted using 18 different real world data sets with various sizes and data distributions. We evaluated all models with 5 different metrics, namely AUC, Specificity, Sensitivity (Recall), F-measure and G-mean. All of the ensemble algorithms in our comparison use non-pruning C4.5 as the base learner. Among all of the algorithms presented for comparison, our proposal shows good performance and has a rather high rank on all metrics, especially for AUC and G-mean, which indicates that our proposal has a good trade-off between the positive classification accuracy and the negative accuracy.

Future work will focus on four aspects. First, Adaptive EUSBoost will be further evaluated on more application domains including bio-informatics and biomedical engineering. Second, because the undersampling process is likely to cause the problem of data underrepresentation, it is reasonable to try out some alternative classifiers, such as Support Vector Machine (SVM), as the base classifier. Third, the method for finding the optimal decision boundary is still needs in need of improvement because the currently used OTSU is time consuming and, on the other hand, the fuzzy threshold is worthy of further research. Last, we will attempt to extend our proposal to multi-class classification tasks. The EUS procedure can be executed on each class to obtain several independent subsets, and these subsets can be used for building sub-classifiers. Then, multi-class classification techniques such as one-vs-one, one-against-all, and error-coding based methods can take the place of binary C4.5 for classification.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61271069).

The authors would like to thank Zheng Tong for reviewing the complexity computation of our proposal. We would also like to thank the anonymous reviewers and the Associate Editor for the constructive evaluation of this paper.

References

Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F., 2010. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J. Multiple-Valued Log. Soft Comput. 17 (2–3), 255–287.

Bao, L., Juan, C., Li, J., Zhang, Y., 2016. Boosted near-miss under-sampling on SVM ensembles for concept detection in large-scale imbalanced datasets. Neurocomputing 172, 198–206.

Barandela, R., Valdovinos, R.M., Sánchez, J.S., 2003. New applications of ensembles of classifiers. Pattern Anal. Appl. 6 (3), 245–256.

Barua, S., Islam, M.M., Yao, X., Murase, K., 2014. MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning. IEEE Trans. Knowl. Data Eng. 26 (2), 405–425.

Breiman, L., 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. J.Artificial Intell.Res. 16, 321–357.

Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W., 2003. SMOTEBoost: improving prediction of the minority class in boosting. In: European Conference on Principles of Data Mining and Knowledge Discovery. Springer, pp. 107–119.

Dal Pozzolo, A., Caelen, O., Le Borgne, Y.-A., Waterschoot, S., Bontempi, G., 2014. Learned lessons in credit card fraud detection from a practitioner perspective. Expert Syst. Appl. 41 (10), 4915–4928.

Drummond, C., Holte, R.C., et al., 2003. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In: Workshop on Learning From Imbalanced Datasets II, 11. Citeseer, pp. 1–8.

Elkan, C., 2001. The foundations of cost-sensitive learning. In: International Joint Conference on Artificial Intelligence, 17. LawrenceErlbaum Associates LTD, pp. 973–978.

Ertekin, S., Huang, J., Bottou, L., Giles, L., 2007. Learning on the border: active learning in imbalanced data classification. In: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management. ACM, pp. 127–136.

Fan, W., Stolfo, S.J., Zhang, J., Chan, P.K., 1999. AdaCost: misclassification cost-sensitive boosting. In: ICML, pp. 97–105.

Fawcett, T., 2004. ROC graphs: notes and practical considerations for researchers. Mach. Learn. 31, 1–38.

Fawcett, T., 2006. An introduction to ROC analysis. Pattern Recognit. Lett. 27 (8), 861–874.

Fernández, A., del Jesus, M.J., Herrera, F., 2015. Addressing overlapping in classification with imbalanced datasets: afirst multi-objective approach for feature and instance selection. In: Intelligent Data Engineering and Automated Learning–IDEAL 2015. Springer, pp. 36–44.

Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55 (1), 119–139.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., Herrera, F., 2012. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans. Syst., Man, Cybern., Part C, 42 (4), 463–484.

Galar, M., Fernández, A., Barrenechea, E., Herrera, F., 2013. EUSBoost: enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. Pattern Recognit. 46 (12), 3460–3471.

Geman, S., Bienenstock, E., Doursat, R., 1992. Neural networks and the bias/variance dilemma. Neural Comput. 4 (1), 1–58.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009. The WEKA data mining software: an update. ACM SIGKDD Explor. Newslett. 11 (1), 10–18.

Han, H., Wang, W.-Y., Mao, B.-H., 2005. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: International Conference on Intelligent Computing. Springer, pp. 878–887.

He, H., Bai, Y., Garcia, E.A., Li, S., 2008. 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.

He, H., Garcia, E.A., 2009. Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. 21 (9), 1263–1284.

Hong, X., Chen, S., Harris, C.J., 2007. A kernel-based two-class classifier for imbalanced data sets. IEEE Trans. Neural Networks 18 (1), 28–41.

Kang, P., Cho, S., 2006. EUS SVMs: ensemble of under-sampled SVMs for data imbalance problems. In: Neural Information Processing. Springer, pp. 837–846.

Krawczyk, B., Woźniak, M., Schaefer, G., 2014. Cost-sensitive decision tree ensembles for effective imbalanced classification. Appl. Soft Comput. 14, 554–562.

Laradji, I.H., Alshayeb, M., Ghouti, L., 2015. Software defect prediction using ensemble learning on selected features. Inf Softw Technol 58, 388–402.

Lee, H.-j., Cho, S., 2006. The novelty detection approach for different degrees of class imbalance. In: International Conference on Neural Information Processing. Springer, pp. 21–30.

Li, K., Fang, X., Zhai, J., Lu, Q., 2016. An imbalanced data classification method driven by boundary samples-Boundary-Boost. In: Information Science and Control Engineering (ICISCE), 2016 3rd International Conference on. IEEE, pp. 194–199.

Liu, T.-Y., 2009. Easyensemble and feature selection for imbalance data sets. In: Bioinformatics, Systems Biology and Intelligent Computing, 2009. IJCBS'09. International Joint Conference on. IEEE, pp. 517–520.

Lobo, J.M., Jiménez-Valverde, A., Real, R., 2008. AUC: a misleading measure of the performance of predictive distribution models. Global Ecol. Biogeogr. 17 (2), 145–151.

Manevitz, L., Yousef, M., 2007. One-class document classification via neural networks. Neurocomputing 70 (7), 1466–1481.

Menardi, G., Torelli, N., 2014. Training and assessing classification rules with imbalanced data. Data Min. Knowl. Discov. 28 (1), 92–122.

Mohd Pozi, M.S., Sulaiman, M.N., Mustapha, N., Perumal, T., 2015. A new classification model for a class imbalanced data set using genetic programming and support vector machines: case study for wilt disease classification. Remote Sens. Lett. 6 (7), 568–577.

Otsu, N., 1975. A threshold selection method from gray-level histograms. Automatica 11 (285–296), 23–27.

Parvin, H., Minaei-Bidgoli, B., Alinejad-Rokny, H., 2013. A new imbalanced learning and dictions tree method for breast cancer diagnosis. J. Bionanosci. 7 (6), 673–678.

Quinlan, J.R., 1993. C4. 5: Programming for Machine Learning. Morgan Kauffmann.

Quinlan, J.R., 2014. C4. 5: Programs for machine learning. Elsevier.

Schapire, R.E., Singer, Y., 1999. Improved boosting algorithms using confidence-rated predictions. Mach. Learn. 37 (3), 297–336.

Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J., Napolitano, A., 2010. RUSBoost: a hybrid approach to alleviating class imbalance. IEEE Trans. Syst., Man Cybern., Part A 40 (1), 185–197.
Sun, Y., Kamel, M.S., Wong, A.K., Wang, Y., 2007. Cost-sensitive boosting for classification of imbalanced data. Pattern Recognit. 40 (12), 3358–3378.
Tang, Y., Zhang, Y.-Q., 2006. Granular SVM with repetitive undersampling for highly imbalanced protein homology prediction. In: 2006 IEEE International Conference on Granular Computing. IEEE, pp. 457–460.
Wang, S., Yao, X., 2009. Diversity analysis on imbalanced data sets by using ensemble models. In: Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on. IEEE, pp. 324–331.
Webb, G.I., 2000. Multiboosting: a technique for combining boosting and wagging. Mach. Learn. 40 (2), 159–196.
Wu, X., Kumar, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Philip, S.Y., et al., 2008. Top 10 algorithms in data mining. Knowl. Inf. Syst. 14 (1), 1–37.
Yang, Q., Wu, X., 2006. 10 Challenging problems in data mining research. Int. J. Inf. Technol. Decis. Mak. 5 (04), 597–604.
Yousefian-Jazi, A., Ryu, J.-H., Yoon, S., Liu, J.J., 2014. Decision support in machine vision system for monitoring of TFT-LCD glass substrates manufacturing. J. Process Control 24 (6), 1015–1023.
Zhang, J., Wang, K., Zhu, W., Zhong, P., 2015. Least squares fuzzy one-class support vector machine for imbalanced data. Int. J. Signal Process., Image Process. Pattern Recognit. 8 (8), 299–308.
Zhang, Y., Zhang, D., Mi, G., Ma, D., Li, G., Guo, Y., Li, M., Zhu, M., 2012. Using ensemble methods to deal with imbalanced data in predicting protein–protein interactions. Comput. Biol. Chem. 36, 36–41.

Wei Lu received his B.Eng. degree in Electronic Engineering, and Ph.D. degree in signal and information processing from Tianjin University, Tianjin, China, in 1998 and 2003 respectively. He is currently an associate professor in the Schoolof Electronic Information Engineering, Tianjin University. His teaching and research interests include digital filter design, digital multimedia technology, embedded system design, Web application design, and pattern recognition. He is now a senior memberof the Chinese Institute of Electronics.

Zhe Li received his B.Eng. degree in Information and Communication Engineering from Zhejiang University, Hangzhou, China in 2014. He is now a master candidate in the School of Electronic Information Engineering in Tianjin University, Tianjin, China. His research involves machine learning, data mining, pattern recognition, and digital image processing.

Jinghui Chu received the B.Eng. degree in radio technology, and M.Eng. and Ph.D. degrees in signal and information processing all from Tianjin University, Tianjin, China, in 1991, 1997, and 2006 respectively. She is currently an associate professor in the School of Electronic Information Engineering, Tianjin University. Her teaching and research interests include digital video technology and pattern recognition.