Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 761814, 12 pages
http://dx.doi.org/10.1155/2013/761814

Research Article

An Empirical Study on the Performance of Cost-Sensitive Boosting Algorithms with Different Levels of Class Imbalance

Qing-Yan Yin, Jiang-She Zhang, Chun-Xia Zhang, and Sheng-Cai Liu

School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, China

Correspondence should be addressed to Jiang-She Zhang; jszhang@mail.xjtu.edu.cn

Received 10 May 2013; Accepted 13 August 2013

Academic Editor: Reza Jazar

Cost-sensitive boosting algorithms have proven successful for solving the difficult class imbalance problems. However, the influence of misclassification costs and imbalance level on the algorithm performance is still not clear. The present paper aims to conduct an empirical comparison of six representative cost-sensitive boosting algorithms, including AdaCost, CSB1, CSB2, AdaC1, AdaC2, and AdaC3. These algorithms are thoroughly evaluated by a comprehensive suite of experiments, in which nearly fifty thousands classification models are trained on 17 real-world imbalanced data sets. Experimental results show that AdaC serial algorithms generally outperform AdaCost and CSB when dealing with different imbalance level data sets. Furthermore, the optimality of AdaC2 algorithm stands out around the misclassification costs setting: $C_N = 0.7$, $C_P = 1$, especially for dealing with strongly imbalanced data sets. In the case of data sets with a low-level imbalance, there is no significant difference between the AdaC serial algorithms. In addition, the results indicate that AdaC1 is comparatively insensitive to the misclassification costs, which is consistent with the finding of the preceding research work.

1. Introduction

Classification is an important task of knowledge discovery and data mining. A large number of classification algorithms have been well developed, such as decision tree, neural network, the Bayesian network, and support vector machine. These algorithms always assume a relatively balanced class distribution. However, class imbalance problems are frequently encountered in many real-world applications including medical diagnosis [1], fraud detection [2], fault diagnosis [3], text categorization [4], and DNA sequences analysis [5]. The class imbalance problem has emerged as an intractable issue due to the difficulty caused by the imbalanced class distribution.

The imbalanced class distribution is characterized as having many more instances of some classes than others. Particularly for the two-class task that we consider in this paper, it occurs when samples of the majority class representing the negative concept outnumber samples of the minority class representing the positive concept. It has been reported that conventional classifiers exhibit serious performance degradation for class imbalance problems, since they show a strong bias towards the majority class. However, the correct classification of the minority class is more preferred than the majority class. For example, the recognition goal is to provide a higher identification rate of rare diseases in medical diagnosis.

In view of the importance of this issue, a great deal of research work has been carried out in recent years [6–8]. The main research can be categorized into three groups. The first group focuses on the approaches for handling the imbalance both at the data and algorithm levels. The second group explores proper evaluation metrics for imbalanced learning algorithms [9, 10]. The third one is to study the nature of the class imbalance problem; that is, what data characteristics aggravate the problem, and whether there are other factors that lead to performance reduction of classifiers [11, 12].

Data level techniques add a preprocessing step to rebalance the class distribution by resampling the data space, including oversampling positive instances and undersampling negative instances [13, 14]. There are also some methods that involve a combination of the two sampling

methods [15, 16]. When discussing what is the best data level solution for this issue, Van Hulse et al. [17] suggested that the utility of each particular resampling strategy depends on various factors, including the imbalance ratio, the characteristics of data, and the nature of classifier. Recently, García et al. [18] significantly extended previous works by deeply investigating the influences of the imbalance ratio and the classifier on the effectiveness of the most popular resampling strategies. Their experimental results showed that oversampling consistently outperforms undersampling for strongly imbalanced data sets, whereas there are no significant differences for data sets with a low-level imbalance.

At the algorithm level, the objective is to adapt existing learning algorithms to bias towards the minority class. These methods require special knowledge of both the corresponding classifier and the application domain [19, 20]. In addition, cost-sensitive learning algorithms fall between data and algorithm level approaches. They incorporate both data level transformations (by adding costs to instances) and algorithm level modifications (by modifying the learning process to accept costs). Interested readers can refer to the relevant literature [21–26].

In recent years, ensemble-based learning algorithms have arisen as a group of popular methods for solving class imbalance problems. The modification of the ensemble learning algorithm includes data level approaches to preprocess the data before learning each classifier [27, 28]. Besides, some proposals consider the embedding of the cost-sensitive framework in the ensemble learning process, which is also known as cost-sensitive boosting [29–34]. For this kind of algorithms, the proper misclassification costs are essential for their good performance. When handling imbalanced classification problems, the imbalance level of the data set will undoubtedly impact the optimal misclassification costs of these algorithms. To the best of our knowledge, it is still not clear how the misclassification costs and imbalance level affect the performance of these cost-sensitive boosting algorithms so far.

Motivated by the previous analysis, we made a thorough empirical study to investigate the effect of both imbalance level and misclassification costs on the performance of some popular cost-sensitive boosting algorithms, including AdaCost [29], CSB1, CSB2 [30], and AdaC serial algorithms (AdaC1, AdaC2, and AdaC3) [31]. To this end, we carry out a comprehensive suite of experiments by employing 17 real-world data sets, four performance metrics and fifty thousands training models, providing a complete perspective on the performance evaluation. The comparison results are tested for statistical significance via the *paired t-test* and visualized by the multidimensional scaling analysis.

The rest of this paper is organized as follows. Section 2 reviews several cost-sensitive boosting algorithms based on AdaBoost. In Section 3, we describe the experimental framework, including experimental data sets, cost setups, performance measures, and experimental approaches. In Section 4, we discuss experimental results to obtain some valuable findings. Finally, conclusions and some future work are outlined in Section 5.

2. AdaBoost and Its Cost-Sensitive Modifications

Ensemble methods have emerged as meta-techniques for improving the generalization performance of existing learning algorithms. The basic idea is to construct several classifiers from the original data and then combine them to obtain a new classifier that outperforms each one of them. Boosting and bagging are the most widely used ensemble learning algorithms, which have led to significant improvements in many real-world applications.

2.1. AdaBoost Algorithm. As the first applicable approach, AdaBoost [35] has been the most representative algorithm in the family of boosting. In particular, AdaBoost has been appointed as one of the top ten data mining algorithms [36].

AdaBoost uses the whole data set to train base classifiers serially and gives each sample a weight reflecting its importance. At the end of each iteration, the weight vector is adjusted so that the weights of misclassified instances are increased and those of correctly classified ones are decreased. Furthermore, another weight vector is assigned to individual classifiers depending on their accuracy. When a test instance is submitted, each classifier gives a weighted vote, and the final predicted class label is selected by majority. The pseudocode for AdaBoost algorithm is shown in Algorithm 1.

The sample weighting strategy of AdaBoost is equivalent to resampling the data space combining both undersampling and over-sampling. When dealing with imbalanced data sets, AdaBoost tends to improve the identification accuracy of the positive class since it focuses on misclassified samples. Hence, it makes the AdaBoost an attractive algorithm for class imbalance problems.

2.2. Cost-Sensitive Boosting Algorithms. However, since AdaBoost is an accuracy-oriented algorithm, so the learning strategy may bias towards the negative class as it contributes more to the overall accuracy. Moreover, reported works show that the improved identification performance on the positive class is not always satisfactory. Hence, it requires AdaBoost algorithm to adapt its boosting strategy towards the cost-sensitive learning framework.

Cost-sensitive learning assigns different costs to different types of misclassification. Let $C(i, j)$ denote the cost of predicting an example from class i as class j. For the two-class case, the cost of misclassifying a positive instance is denoted by C_P, and the contrary one is denoted by C_N. The recognition importance of positive instances is higher than that of negative instances. Hence, the cost of misclassifying the positive class is greater than that of the negative class; that is, $C_P > C_N$. The cost-sensitive learning adds the cost matrix into the model building process and generates a model that minimizes the total misclassification cost.

In present paper, we focus on several representative algorithms in the family of cost-sensitive boosting, including AdaCost [29], CSB1, CSB2 [30], and AdaC serial algorithms (AdaC1, AdaC2, and AdaC3) [31]. They differ in the way of how to introduce cost items into the weighted distribution D^t in the AdaBoost framework.

(i) **Input**: training set $(x_1, y_1), \ldots, (x_N, y_N)$, where $x_i \in X$, $y_i \in Y = \{-1, +1\}$, BaseLearn algorithm, Number of iterations T.
(ii) **Initialization**: the weighted distribution of training samples: $D^1(i) = 1/N,\ i = 1, \ldots, N$.
(iii) **Iteration**: For $t = 1, \ldots, T$:
(1) Use the BaseLearn algorithm to train a component classifier h_t on the training data set sampling from distribution D^t.
(2) Calculate the training error ε_t of the classifier h_t: $\varepsilon_t = \sum_{i:y_i \neq h_t(x_i)} D^t(i)$
(3) Set the weight α_t for the classifier h_t: $\alpha_t = (1/2)\log(1 - \varepsilon_t)/\varepsilon_t$.
(4) Update the weighted distribution of training samples:

$$D^{t+1}(i) = \frac{D^t(i) \cdot e^{-\alpha_t h_t(x_i) y_i}}{Z_t},$$

where Z_t is the normalization constant so that D^{t+1} will be a distribution, that is,

$$Z_t = \sum_i D^t(i) \cdot e^{-\alpha_t h_t(x_i) y_i}.$$

(iv) **Output**: The final hypothesis:

$$H(x) = \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Algorithm 1: AdaBoost algorithm.

2.2.1. AdaCost. In this algorithm, the weight update rule increases the weights of misclassified samples more aggressively but decreases the weights of correctly classified samples more conservatively. This is accomplished by introducing the cost adjustment function β into the weight update formula as follows:

$$D^{t+1}(i) = \frac{D^t(i) \cdot e^{-\alpha_t h_t(x_i) y_i \beta_{\operatorname{sgn}(h_t(x_i), y_i)}}}{Z_t}, \tag{1}$$

where $\operatorname{sgn}(h_t(x_i), y_i)$ denotes "+" when $h_t(x_i)$ equals y_i; that is, x_i is correctly classified, "−" otherwise. Fan et al. [29] provided the recommended setting:

$$\beta_+ = -0.5C_i + 0.5, \qquad \beta_- = 0.5C_i + 0.5, \tag{2}$$

where C_i is the cost of misclassifying the ith example and β_+ (β_-) denotes the output of correctly (incorrectly) classified samples, respectively. Since $C_P > C_N$, then we have $\beta_+^+ < \beta_+^-$ and $\beta_-^+ > \beta_-^-$. Hence, false negative receives greater weight increase than false positive, and true positive loses less weight than true negative.

The weight updating parameter α_t is computed as

$$\alpha_t = \frac{1}{2}\log\frac{1 + r_t}{1 - r_t}, \tag{3}$$

where

$$r_t = \sum_i D^t(i)\, y_i h_t(x_i)\, \beta_{\operatorname{sgn}(h_t(x_i), y_i)}. \tag{4}$$

2.2.2. CSB1 and CSB2. CSB1 modifies the weight update formula of AdaBoost to

$$D^{t+1}(i) = \frac{D^t(i) \cdot C_{\operatorname{sgn}(h_t(x_i), y_i)} e^{-h_t(x_i) y_i}}{Z_t}. \tag{5}$$

And CSB2 changes it to

$$D^{t+1}(i) = \frac{D^t(i) \cdot C_{\operatorname{sgn}(h_t(x_i), y_i)} e^{-\alpha_t h_t(x_i) y_i}}{Z_t}. \tag{6}$$

The difference between CSB1 and CSB2 mainly lies in weight parameter α_t: CSB1 does not use any α_t factor, and CSB2 updates α_t in the same way as that of AdaBoost. Even though the weight update formula of CSB2 is similar to that of AdaC2, CSB2 does not take cost items into consideration in the update rule of parameter α_t in the learning process.

2.2.3. AdaC1. This algorithm is one of the three cost-sensitive modifications of AdaBoost proposed by Sun et al. [31]. These three algorithms derive different weighted distribution update formulas depending on where they introduce cost items. In AdaC1, cost items are embedded inside the exponent part of weight update formula:

$$D^{t+1}(i) = \frac{D^t(i) \cdot e^{-\alpha_t C_i h_t(x_i) y_i}}{Z_t}, \tag{7}$$

where $C_i \subset [0, +\infty]$ is an associated cost item with the ith sample. The weight parameter α_t can be induced in the similar way as AdaBoost:

$$\alpha_t = \frac{1}{2}\log\frac{1 + \sum_{i:y_i = h_t(x_i)} C_i D^t(i) - \sum_{i:y_i \neq h_t(x_i)} C_i D^t(i)}{1 - \sum_{i:y_i = h_t(x_i)} C_i D^t(i) - \sum_{i:y_i \neq h_t(x_i)} C_i D^t(i)}. \tag{8}$$

Note that AdaCost is a variation of AdaC1 by introducing the cost adjustment function instead of cost items inside the exponent part. All the AdaC serial algorithms can be reduced to AdaBoost when all the cost items are equally set to 1, but AdaCost cannot be reduced to AdaBoost.

2.2.4. AdaC2. Unlike AdaC1, AdaC2 embeds cost items in a different way that is outside the exponent part:

$$D^{t+1}(i) = \frac{C_i D^t(i) \cdot e^{-\alpha_t h_t(x_i) y_i}}{Z_t}. \tag{9}$$

Accordingly, the computation of the parameter α_t is changed to

$$\alpha_t = \frac{1}{2} \log \frac{\sum_{i:y_i = h_t(x_i)} C_i D^t(i)}{\sum_{i:y_i \neq h_t(x_i)} C_i D^t(i)}. \tag{10}$$

2.2.5. AdaC3. This modification combines the idea of AdaC1 and AdaC2 simultaneously. Namely, the weight update formula is modified by introducing cost items both inside and outside the exponent part:

$$D^{t+1}(i) = \frac{C_i D^t(i) \cdot e^{-\alpha_t C_i h_t(x_i) y_i}}{Z_t}. \tag{11}$$

Thereby, the weight parameter α_t is computed as follows:

$$\begin{aligned} \alpha_t = \frac{1}{2} \log \Bigg(& \Bigg(\sum_i C_i D^t(i) + \sum_{i:y_i = h_t(x_i)} C_i^2 D^t(i) \\ & - \sum_{i:y_i \neq h_t(x_i)} C_i^2 D^t(i) \Bigg) \\ & \times \Bigg(\sum_i C_i D^t(i) - \sum_{i:y_i = h_t(x_i)} C_i^2 D^t(i) \\ & + \sum_{i:y_i \neq h_t(x_i)} C_i^2 D^t(i) \Bigg)^{-1} \Bigg). \end{aligned} \tag{12}$$

3. Experimental Framework

In this section, we present the experimental framework used to carry out the empirical study to evaluate the above cost-sensitive boosting algorithms. The aim of this study is to investigate how the algorithm performance is affected when different cost settings and different imbalance levels are considered in the experiment.

3.1. Experimental Data Sets. In the experiment, we employed 17 public imbalanced data sets from the UCI machine learning repository, which have also been used [18]. The chosen data sets vary in sample size, class distribution, and imbalance ratio in order to ensure a thorough performance assessment. We discussed only the binary classification problems in this paper. Some data sets with multiclass labels were transformed into two-class ones, by keeping one class as the positive class and joining the remaining classes into the negative class.

Table 1 summarizes the properties of the used data sets for each data set, the total number of samples, the indicia, and the sample size of the minority and majority classs. The last column is the imbalance ratio, which is defined as sample size of the majority class divided by that of the minority class. This table is ordered according to the imbalance ratio in the descending order.

The large imbalance ratio means the high-level imbalance, and the small one denotes the low-level imbalance. Experimental data sets were divided into two groups according to the imbalance ratio. The first group is deemed as strongly imbalanced, including LetterA, Cbands, Pendigits, Satimage, Optidigts, Mfeat_kar, Mfeat_zer, Segment, and Scrapie, whose imbalance ratios are larger than 4. The second group consists of the low-level imbalanced data sets: Vehicle, Haberman, Yeast, Breast, Phoneme, German, Pima, and Spambase.

3.2. Cost Setups. In the experiment, we will study the influence of different cost settings on the performance of these cost-sensitive boosting algorithms. In these algorithms, misclassification costs are used to characterize the recognition importance of different samples. The proper misclassification costs are often unavailable and can be ascertained using the empirical method.

In our experiments, the misclassification costs for samples in the same category are set with the same value: C_P denotes misclassification cost of the positive class, and C_N denotes that of the negative class. The ratio between C_P and C_N represents the deviation of the learning importance between two classes. The larger the ratio, the more weighted the sample size of the positive class is boosted to strengthen learning. A range of ratio values are tested to search for the most effective cost setting, and we use the cost setup: $C_P = 1$, and C_N varies from 0.1 to 1 with step 0.1.

When $C_P = C_N = 1$, AdaC1, AdaC2, and AdaC3 are all reduced to the original AdaBoost algorithm. For CSB1 and CSB2, the cost setup is suggested by Ting [30]: if a sample is correctly classified, $C_P = C_N = 1$; otherwise, $C_P > C_N \geq 1$. According to the proposal of [31], we fix the cost factor for false negatives C_P as 1 and the cost setting C_N for true positives, true negatives, and false positives.

3.3. Performance Measures. The evaluation metric plays a crucial role in both the guidance of the classifier modeling and the assessment of the classification performance. Traditionally, the total accuracy is the most commonly used performance metric. However, accuracy is no longer a proper measure in imbalanced domains, since the positive class makes little contribution to the overall accuracy. For example, a classifier that predicts all samples to be negative in a data set with an imbalance ratio value of 9 may lead to erroneous conclusions although it achieves a high accuracy of 90%.

In the confusion matrix, all samples can be categorized into four groups, as described by Table 2.

The accuracy evaluates the effectiveness of a classifier by the percentage of correct predictions:

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}}. \tag{13}$$

Table 1: Summary of characteristics for the used data sets.

Data set	Samples	Minority/majority	No. of min/no. of maj	Imbalance ratio
LetterA	20000	Class 1/rest	789/19211	24.35
Cbands	12000	Class 1/rest	500/11500	23.00
Pendigits	10992	Class 5/rest	1055/9937	9.42
Satimage	6435	Class 4/rest	626/5809	9.28
Optidigts	5620	Class 8/rest	554/5066	9.14
Mfeat_kar	2000	Digit 9/rest	200/1800	9.00
Mfeat_zer	2000	Digit 9/rest	200/1800	9.00
Segment	2310	Class 5/rest	330/1980	6.00
Scrapie	3113	Class 1/class 0	531/2582	4.86
Vehicle	846	van/rest	212/634	2.99
Haberman	306	Class 2/class 1	81/225	2.78
Yeast	1484	Class 2/rest	429/1055	2.46
Breast	336	Class 1/class 0	81/196	2.42
Phoneme	5404	Class 1/class 0	1586/3818	2.41
German	1000	Class 2/class 1	300/700	2.33
Pima	768	Class 1/class 0	268/500	1.87
Spambase	4601	Class 1/class 0	1813/2788	1.54

Table 2: Confusion matrix.

	Positive prediction	Negative prediction
Actually positive	True positive (TP)	False negative (FN)
Actually negative	False positive (FP)	True negative (TN)

When dealing with the class imbalance problem, there are other appropriate metrics instead of accuracy. In particular, we can obtain four metrics from the confusion matrix to measure the classification performance of positive and negative classes independently.

True Positive Rate. The percentage of positive instances correctly classified, also known as *sensitivity* or *recall* in the information retrieval domain, is as follows:

$$\mathrm{TP}_{\mathrm{rate}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{14}$$

True Negative Rate. The percentage of negative instances correctly classified, also known as specificity, is as follows

$$\mathrm{TN}_{\mathrm{rate}} = \frac{\mathrm{TN}}{\mathrm{FP} + \mathrm{TN}}. \tag{15}$$

False Positive Rate. The percentage of negative instances misclassified is as follows:

$$\mathrm{FP}_{\mathrm{rate}} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}. \tag{16}$$

False Negative Rate. The percentage of positive instances misclassified, is as follows:

$$\mathrm{FN}_{\mathrm{rate}} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}}. \tag{17}$$

On the other hand, in the case that high classification performance on the positive class is demanded, the precision metric is often adopted:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{18}$$

When good quality performance for both classes is required, none of these metrics alone is adequate by itself. Hence, some more complex evaluation measures have been devised.

(1) F-Measure. If only the performance of the positive class is considered, $\mathrm{TP}_{\mathrm{rate}}$ and *precision* are important metrics. *F*-measure integrates these two metrics as follows:

$$F\text{-measure} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}. \tag{19}$$

Evidently, *F*-measure represents the harmonic mean of *precision* and *recall*, and it tends to be closer to the smaller of these two measures. Hence, a higher *F*-measure value ensures that both *recall* and *precision* are reasonably high.

(2) G-Mean. When the performance of both classes is concerned, both $\mathrm{TP}_{\mathrm{rate}}$ and $\mathrm{TN}_{\mathrm{rate}}$ are expected to be high meanwhile. The *G*-mean metric is defined as

$$G\text{-mean} = \sqrt{\mathrm{TP}_{\mathrm{rate}} \cdot \mathrm{TN}_{\mathrm{rate}}}. \tag{20}$$

G-mean represents the geometric mean of $\mathrm{TP}_{\mathrm{rate}}$ and $\mathrm{TN}_{\mathrm{rate}}$, and so, it measures the balanced performance of a learning algorithm between two classes.

3.4. Experimental Approaches. For each data set, we performed 5 independent runs of a stratified 10-fold cross-validation to partition the whole data and obtained 50-dimensional score vector for each algorithm. Moreover, we

had 60 models for each data set, which came from 6 cost-sensitive boosting algorithms (AdaCost, CSB1, CSB2, AdaC1, AdaC2, and AdaC3) and 10 cost settings. For comparison purposes, we also included the best results of these models, and thus got a total number of 61 models in the experiment.

Similar to the statistical analysis method used in [18], we adopt *paired t-test* to determine whether one algorithm is significantly better than another one and then use the multidimensional scaling to visually compare the performance of different cost-sensitive algorithms.

(1) Paired t-Test. Given two paired sets X_i and Y_i of N measured values, the *paired t-test* determines whether they differ from each other significantly under the assumptions that the paired differences are independent and identically normally distributed. In light of the central limit theorem, the sampling distribution of any statistic will be approximately normally distributed, if the sample size N is large enough. As a rough rule of thumb, a sample size of 30 is large enough. In this paper, we conduct the statistical comparison between each pair of 50-dimensional score vectors, which can be regarded as approximately normally distributed, and so, it is reasonable to employ parametric *paired t-test* for statistical comparisons. Based on pairwise comparisons for these algorithms, we computed the *index* of performance as the difference between *wins* and *losses*, where *wins (losses)* denotes the total times that an algorithm has been significantly better (worse) than others with a significance level $\alpha = 0.05$.

(2) Multidimensional Scaling. The second complementary analysis tool is multidimensional scaling (MDS) [37, 38], which aims at giving a visual comparison for classifier performance with respect to multiple metrics. We built a $61 \times D$ table for each performance metric, where D denotes the number of data sets used in the experiment. Each element (i, j) represents the average score of the model i on the data set j, which was calculated by 5 runs of 10-fold cross-validation. Then, we computed Euclidean distances between each pair of rows in the table and performed multidimensional scaling on the distance matrix in order to obtain a projection on 2-dimension space. We can determine the effectiveness of various algorithms through the dispersal trend of their performance scores towards the optimal point in the MDS space.

4. Experimental Results

Aiming to study the influence of the imbalance ratio and misclassification costs on the performance of different cost-sensitive boosting algorithms, we performed both statistical test and MDS analysis for data sets with high-level and low-level imbalances separately.

4.1. Results on Severely Imbalanced Data Sets. First of all, we perform the significance test of different cost-sensitive boosting algorithms to show whether there exist significant differences among them. To this end, we use the *paired t-test* on the combinations of different algorithms and different cost settings.

For each combination, we achieve some 50-dimensional vectors which come from the results of 5 runs 10-fold cross-validation on all the training data sets. Then, we conduct the *paired t-test* between each pair of these score vectors. The *wins* of all the data sets are added into the final *wins*, and the final *losses* is gained likewise. The *index* of performance is calculated as the difference between *wins* and *losses*. Tables 3 and 4 provide the indices of performance for severely imbalanced data sets using F-measure and G-mean metrics, respectively. Note that the best result of each cost setting is signed with **framed boxes**.

From the results of severely imbalanced data sets, we can observe that in most cases, AdaCost is always significantly the firstly worst, CSB1 and CSB2 are secondly worst with negative index values absolutely. Among AdaC serial algorithms, AdaC2 is more preferred than others, especially when using the G-mean metric. From the point of view of F-measure, AdaC2 is slightly better than the others except for the first three cost settings. AdaC serial algorithms have similar performances when $C_N = 1$, since they are all reduced to AdaBoost.

Owing to the obvious advantage of AdaC serial algorithms over the other two groups, we will only include AdaC serial algorithms in the MDS analysis. For each data set, we have 30 models which come from 3 algorithms (AdaC1, AdaC2, and AdaC3) and 10 cost settings. For each model, we obtain the average score as the mean of 50-dimensional score vectors. We build a 31×9 table whose element (i, j) represents the average score of the model i on the data set j. Then, we perform multidimensional scaling on the distance matrix to obtain the projection on 2-dimension space.

Figure 1 illustrates the MDS plots of AdaC serial algorithms on severely imbalanced data sets. The number associated with each point denotes some cost setting. For example, the number 2 with the blue circle denotes the model of AdaC2 at the cost setting $C_N = 0.2, C_P = 1$. We can comprehend the meaning of other points, and so forth.

As we might imagine, when $\mathrm{TP}_{\mathrm{rate}}$ is considered, points of AdaC2 and AdaC3 become more and more close to the optimal point as C_N decreases. Moreover, points of AdaC2 are always closer to the optimal point than those of AdaC3, and points of AdaC1 are much farther than others. Conversely, the $\mathrm{TN}_{\mathrm{rate}}$ apparently presents the opposite behavior. On the other hand, AdaC1 is relatively insensitive to misclassification costs, which is consistent with results of Sun et al. [31].

Focusing on the G-mean metric in Figure 1, we can see that AdaC2 is much better than AdaC3 and AdaC1, which is consistent with the result of Table 4. However, there are little differences between AdaC serial algorithms for the F-measure metric, and the most proper cost of the negative class lies in [0.7, 0.9]. These findings are further confirmed by Table 5.

As a further confirmation of the previous findings, Table 5 reports Euclidean distances to the optimal point in the MDS space on the severely imbalanced data sets. As expected, the closest points with **framed boxes** generally appear in AdaC2 algorithm for both F-measure and G-mean metrics. We also find that AdaC1 is relatively insensitive to the costs settings. Hence, the average results of AdaC1 tend to be lower than the others. However, it is not implied that AdaC1 is superior to the other two algorithms.

Table 3: Index of performance using F-measure for severely imbalanced data sets.

	Win	Loss	Index	Win	Loss	Index	Win	Loss	Index	Win	Loss	Index	Win	Loss	Index
		0.1			0.2			0.3			0.4			0.5	
AdaC1	19	0	**19**	13	0	**13**	10	0	**10**	12	0	**12**	13	0	13
AdaC2	6	7	−1	8	5	3	12	3	9	13	2	11	16	2	**14**
AdaC3	1	12	−11	1	6	−5	7	5	2	9	5	4	10	4	6
AdaCost	1	4	−3	0	7	−7	0	15	−15	0	21	−21	0	25	−25
CSB1	2	2	0	2	3	−1	0	5	−1	0	5	−5	0	5	−5
CSB2	0	4	−4	0	3	−3	0	1	−4	0	1	−1	0	3	−3
		0.6			0.7			0.8			0.9			1	
AdaC1	12	0	12	13	0	13	10	0	10	12	0	12	14	0	**14**
AdaC2	18	1	**17**	17	1	**16**	17	0	**17**	15	0	**15**	14	0	**14**
AdaC3	11	4	7	9	5	4	11	3	8	14	1	13	13	0	13
AdaCost	0	26	−26	0	27	−27	0	27	−27	0	27	−27	1	27	−26
CSB1	0	7	−7	0	5	−5	0	8	−8	1	10	−9	0	15	−15
CSB2	0	3	−3	0	1	−1	0	0	0	0	4	−4	0	0	0

Table 4: Index of performance using G-mean for severely imbalanced data sets.

	Win	Loss	Index	Win	Loss	Index	Win	Loss	Index	Win	Loss	Index	Win	Loss	Index
		0.1			0.2			0.3			0.4			0.5	
AdaC1	7	0	7	9	0	9	8	0	8	10	0	10	12	0	12
AdaC2	23	1	**22**	31	1	**30**	30	1	**29**	30	1	**29**	28	1	**27**
AdaC3	13	5	8	16	8	8	18	8	10	14	8	6	14	8	6
AdaCost	1	15	−14	0	21	−21	0	20	−20	0	22	−22	0	25	−25
CSB1	1	15	−14	1	15	−14	1	16	−15	1	14	−13	1	12	−11
CSB2	0	9	−9	0	12	−12	0	12	−12	0	10	−10	0	9	−9
		0.6			0.7			0.8			0.9			1	
AdaC1	12	0	12	14	0	14	15	0	15	15	0	15	15	0	**15**
AdaC2	24	1	**23**	26	1	**25**	24	1	**23**	17	1	**16**	15	0	**15**
AdaC3	14	5	9	13	7	6	13	5	8	17	2	15	15	0	**15**
AdaCost	0	26	−26	0	27	−27	0	27	−27	0	27	−27	0	27	−27
CSB1	1	12	−11	0	14	−14	0	15	−15	0	16	−16	0	18	−18
CSB2	0	7	−7	0	4	−4	0	4	−4	0	3	−3	0	0	0

Experimental results show that AdaC serial algorithms are much more better than the other two groups, and AdaC2 is more qualified for handling severely imbalanced classification, especially when the cost of the negative class lies in $[0.7, 0.9]$.

4.2. Results on Data Sets with a Low-Level Imbalance. Similarly, we perform the significance test of different cost-sensitive boosting algorithms for the data sets with a low-level imbalance.

Tables 6 and 7 provide the indices of performance for the F-measure and G-mean metrics, respectively. As can be seen, both AdaCost and CSB serial algorithms are inferior to AdaC serial algorithms, which is very similar to the case of the strongly imbalanced data sets. However, when comparing AdaC serial algorithms, it seems that the differences between them are marginal in the sense that they generally achieve similar indices of performance, especially unlike the outstanding behavior of AdaC2 for severely imbalanced data sets.

Figure 2 illustrates the MDS plots of AdaC serial algorithms on the slightly imbalanced data sets over four evaluation measures. The results for the $\mathrm{TP}_{\mathrm{rate}}$ and $\mathrm{TN}_{\mathrm{rate}}$ are very similar to the case of the strongly imbalanced data sets: AdaC2 and AdaC3 become closer to the optimal $\mathrm{TP}_{\mathrm{rate}}$ as C_N decreases, whereas they are nearer to the optimal $\mathrm{TN}_{\mathrm{rate}}$ as C_N increases. AdaC1 is relatively insensitive to misclassification costs similar to the case of strongly imbalanced data sets.

When analyzing the results of G-mean and F-measure, we can observe that the trend of AdaC2 along with the change of C_N is very similar to that of AdaC3. Moreover, most of the best performances are achieved by AdaC2 algorithm around $C_N = 0.7$. These findings are also clear in Table 8.

From the previous analysis on the data sets with a low-level imbalance, we can conclude that AdaC serial algorithms consistently outperform the other two groups, but it is difficult to advise the best strategy among AdaC1, AdaC2, and AdaC3. It means that the effectiveness of a particular cost-sensitive boosting algorithm depends on the class imbalance

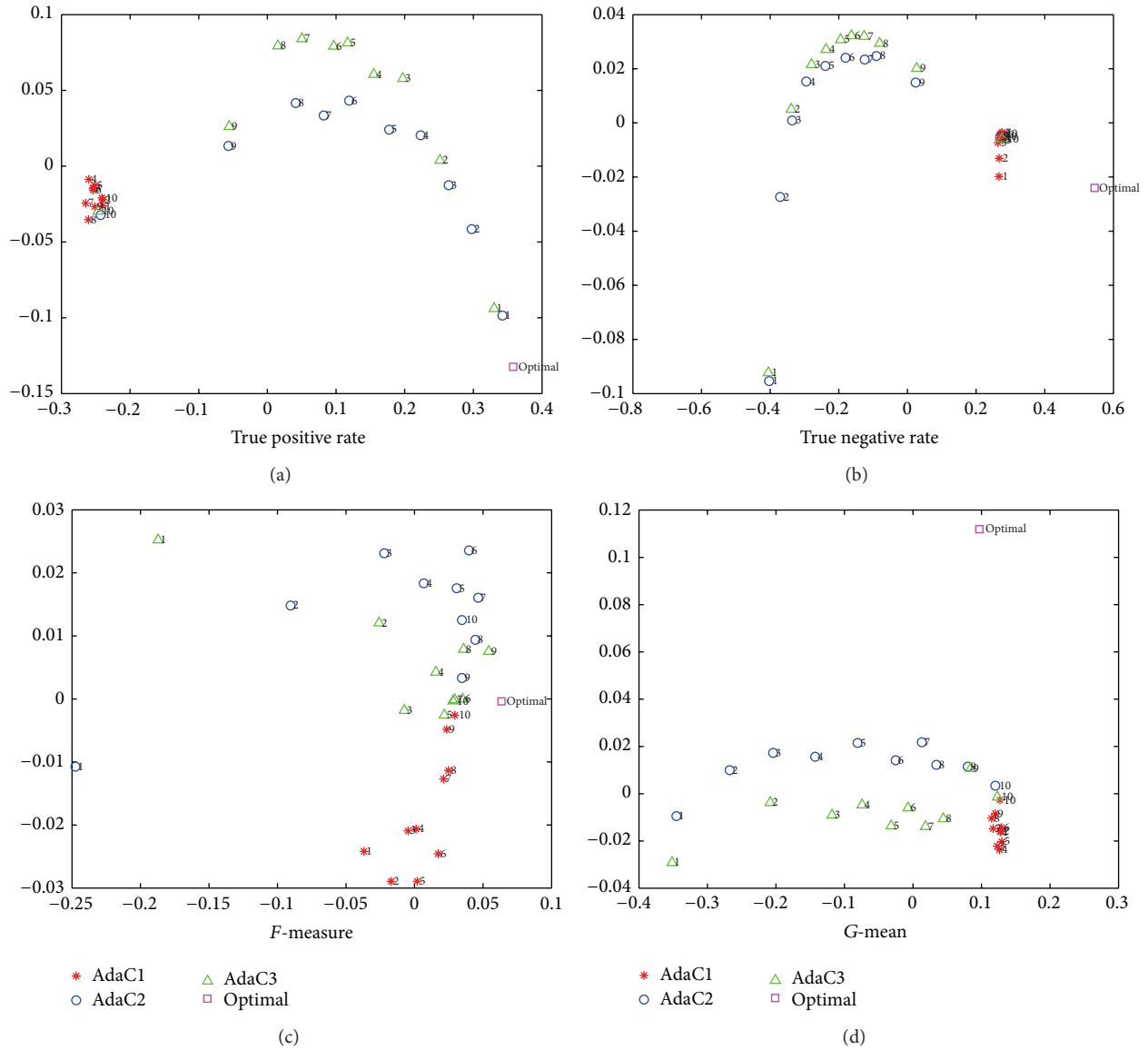

Figure 1: MDS plots of severely imbalanced data sets over four performance metrics.

Table 5: Euclidean distances to the optimal point in the MDS space for data sets with a high imbalance.

C_n	TP_{rate}			F-measure			G-mean		
	AdaC1	AdaC2	AdaC3	AdaC1	AdaC2	AdaC3	AdaC1	AdaC2	AdaC3
0.1	0.6077	**0.0441**	0.0899	0.1058	0.3134	0.2584	0.1324	0.4599	0.4702
0.2	0.6087	0.1140	0.1787	0.0897	0.1565	0.1017	0.1325	0.3803	0.3269
0.3	0.6238	0.1541	0.2502	0.0765	0.0911	0.0809	0.1377	0.3176	0.2477
0.4	0.6308	0.2058	0.2807	0.0687	0.0636	0.0606	0.1402	0.2615	0.2086
0.5	0.6219	0.2403	0.3238	0.0710	0.0441	0.0561	0.1370	0.2025	0.1810
0.6	0.6228	0.2975	0.3366	0.0554	0.0394	0.0452	0.1328	0.1617	0.1605
0.7	0.6321	0.3232	0.3765	0.0504	**0.0297**	0.0499	0.1314	0.1291	0.1513
0.8	0.6265	0.3615	0.4028	0.0497	0.0299	0.0419	0.1263	0.1226	0.1391
0.9	0.6185	0.4398	0.4436	0.0451	0.0387	0.0302	0.1269	**0.1061**	0.1090
1	0.6102	0.6110	0.6139	0.0357	0.0328	0.0364	0.1264	0.1221	0.1239
Av.	0.6203	0.2791	0.3297	0.0648	0.0839	0.0761	0.1324	0.2264	0.2118

Table 6: Index of performance using F-measure for data sets with a low-level imbalance.

	Win	Loss	Index	Win	Loss	Index	Win	Loss	Index	Win	Loss	Index	Win	Loss	Index
		0.1			0.2			0.3			0.4			0.5	
AdaC1	16	0	**16**	10	0	**10**	7	0	**7**	7	0	7	5	0	5
AdaC2	5	5	0	2	4	−2	2	3	−1	9	2	7	11	2	9
AdaC3	1	9	−8	1	5	−4	3	2	1	12	1	**11**	15	0	**15**
AdaCost	5	2	3	2	2	0	2	4	−2	3	7	−4	1	11	−10
CSB1	3	3	0	2	2	0	2	3	−1	2	12	−10	0	12	−12
CSB2	0	11	−11	0	4	−4	0	4	−4	0	11	−11	0	7	−7
		0.6			0.7			0.8			0.9			1	
AdaC1	10	0	10	9	0	9	10	0	10	11	0	11	10	0	10
AdaC2	15	1	14	16	1	15	17	2	15	13	0	**13**	13	0	**13**
AdaC3	18	0	**18**	20	0	**20**	17	0	**17**	13	0	**13**	10	0	10
AdaCost	0	20	−20	0	22	−22	1	23	−22	0	21	−21	1	21	−20
CSB1	0	12	−12	1	11	−10	0	14	−14	0	12	−12	1	11	−10
CSB2	0	10	−10	0	12	−12	0	6	−6	0	4	−4	0	3	−3

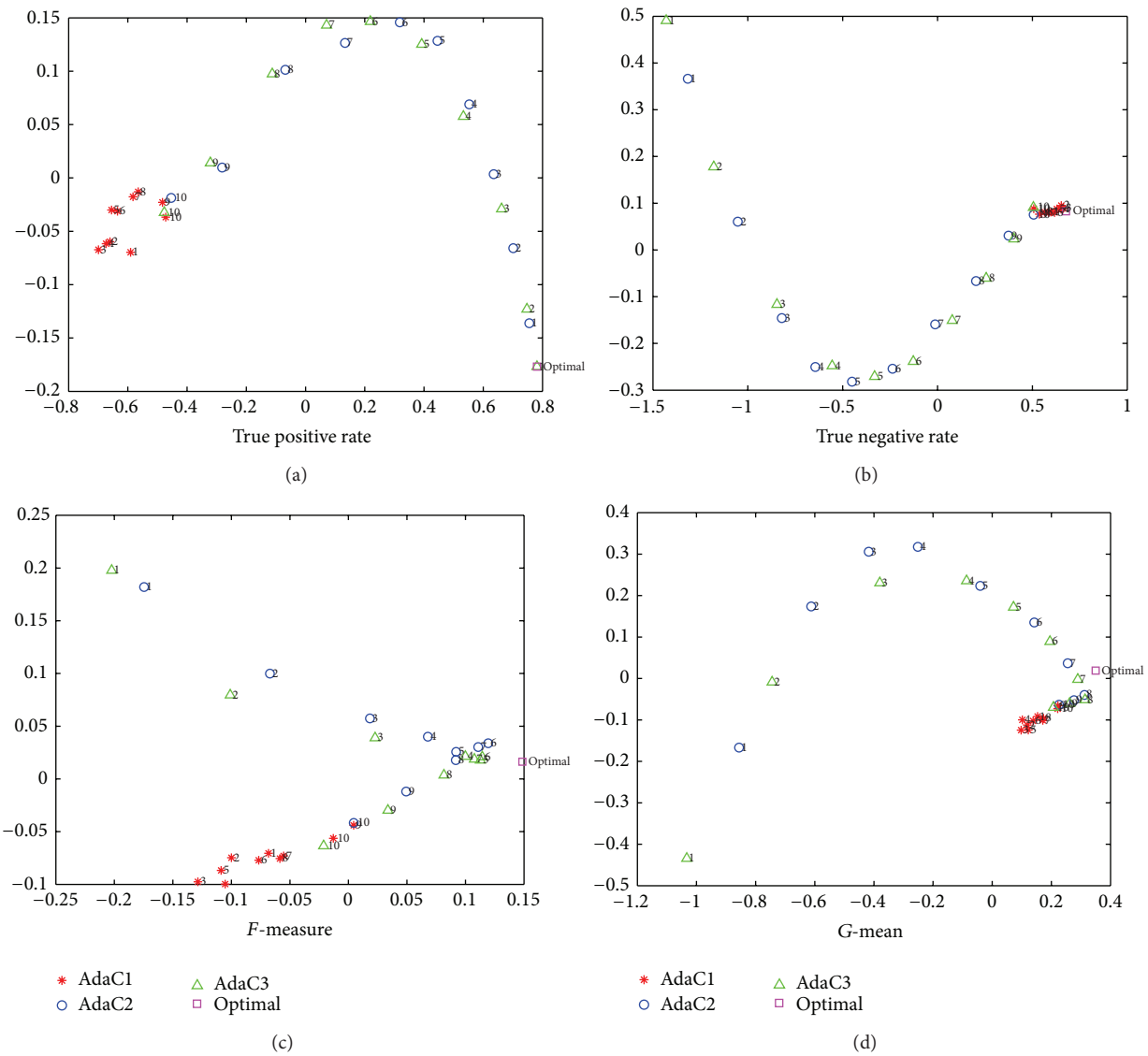

Figure 2: MDS plots for data sets with low-level imbalance over four performance metrics.

Table 7: Index of performance using G-mean for data sets with a low-level imbalance.

	Win	Loss	Index	Win	Loss	Index	Win	Loss	Index	Win	Loss	Index	Win	Loss	Index
		0.1			0.2			0.3			0.4			0.5	
AdaC1	22	0	**22**	18	0	**18**	13	0	**13**	8	0	**8**	6	0	6
AdaC2	6	8	−2	3	7	−4	4	7	−3	3	4	−1	9	2	7
AdaC3	0	14	−14	0	11	−11	2	6	−4	10	2	**8**	15	1	**14**
AdaCost	6	1	5	3	0	3	2	2	0	1	6	−5	0	12	−12
CSB1	7	0	7	3	2	1	0	5	−5	0	6	−6	0	8	−8
CSB2	0	18	−18	0	7	−7	0	1	−1	0	4	−4	0	7	−7
		0.6			0.7			0.8			0.9			1	
AdaC1	12	0	12	12	0	12	10	0	10	11	0	11	15	0	**15**
AdaC2	17	1	16	19	1	18	19	0	**19**	20	0	**20**	15	0	**15**
AdaC3	18	1	**17**	21	2	**19**	19	0	**19**	17	1	16	13	0	13
AdaCost	0	23	−23	0	24	−24	0	24	−24	0	24	−24	0	24	−24
CSB1	0	13	−13	0	15	−15	0	16	−16	0	17	−17	1	17	−16
CSB2	0	9	−9	0	10	−10	0	8	−8	0	6	−6	0	3	−3

Table 8: Euclidean distances to the optimal point in the MDS space for data sets with a low imbalance.

C_n	TP_{rate}			F-measure			G-mean		
	AdaC1	AdaC2	AdaC3	AdaC1	AdaC2	AdaC3	AdaC1	AdaC2	AdaC3
0.1	1.3766	0.0545	**3.5e − 14**	0.2382	0.3645	0.3974	0.2552	1.2215	1.4595
0.2	1.4475	0.1459	0.0778	0.2709	0.2350	0.2635	0.2967	0.9777	1.1015
0.3	1.4853	0.2366	0.1994	0.3021	0.1491	0.1422	0.3265	0.8301	0.7648
0.4	1.4593	0.3389	0.3450	0.2812	0.1055	0.0803	0.3106	0.6788	0.4905
0.5	1.4439	0.4557	0.4935	0.2789	0.0726	0.0548	0.2976	0.4506	0.3288
0.6	1.4239	0.5644	0.6491	0.2471	0.0592	0.0520	0.2669	0.2526	0.1879
0.7	1.3741	0.7167	0.7792	0.2277	**0.0513**	0.0572	0.2357	0.1144	0.0947
0.8	1.3577	0.8941	0.9354	0.2316	0.0700	0.0772	0.2297	**0.0844**	0.0941
0.9	1.2768	1.0818	1.1209	0.1670	0.1098	0.1314	0.1711	0.1099	0.1355
1	1.2640	1.2484	1.2705	0.1841	0.1647	0.1968	0.1754	0.1625	0.1968
Av.	1.3909	0.5737	0.5871	0.2429	0.1382	0.1453	0.2565	0.4883	0.4854

as well as on other factors, such as the data characteristics and the algorithm itself.

5. Conclusions and Future Work

In this paper, we presented a thorough empirical study on the performance of the most popular cost-sensitive boosting algorithms when dealing with different levels of class imbalance. We used 17 real-world imbalanced data sets (9 severely imbalanced and 8 slightly imbalanced), 6 cost-sensitive boosting algorithms (AdaC1, AdaC2, AdaC3, AdaCost, CSB1, and CSB2), and 10 cost settings in the experiment. Besides, the performance of algorithms has been evaluated by means of four different evaluation metrics, that is, TP_{rate}, TN_{rate}, F-measure, and G-mean.

Experimental results show that AdaC serial algorithms (AdaC1, AdaC2, and AdaC3) consistently outperform the other two groups (AdaCost and CSB), both for strongly and slightly imbalanced data sets. Moreover, AdaCost has been demonstrated to be worse than CSB algorithms. When comparing AdaC serial algorithms, AdaC2 is observed to perform better than the two others for severely imbalanced data sets, especially when using the G-mean metric. In the case of data sets with low-level imbalance, however the difference between AdaC serial algorithms is negligible. It is necessary to make a further data complexity analysis to choose a suitable algorithm for a particular imbalanced data set.

On the other hand, we have given some guidance in choosing the proper misclassification costs for these cost-sensitive boosting algorithms. Summarizing the experimental results, it demonstrated that the most proper cost setting is located in the neighbourhood of the point $C_N = 0.7$, $C_P = 1$.

Based on the present work, there are some interesting future research directions with regard to the class imbalance problem: (1) to utilize other parameter selection techniques for the confirmation of the proper misclassification costs; (2) to take other cost-sensitive learning algorithms into consideration within the present framework, such as the proposed algorithms of [22–24, 39]; (3) to compare these cost-sensitive

algorithms in terms of other performance metrics, such as AUC [9] and IBA [40].

Acknowledgments

This work was supported by the National Basic Research Program of China (973 Program) under Grant no. 2013CB329404, the Major Research Project of the National Natural Science Foundation of China under Grant nos. 91230101, the National Natural Science Foundation of China under Grant no. 61075006 and 11201367, the Key Project of the National Natural Science Foundation of China under Grant no. 11131006, and the Research Fund for the Doctoral Program of Higher Education of China under Grant no. 20100201120048.

References

[1] G. Cohen, M. Hilario, H. Sax, S. Hugonnet, and A. Geissbuhler, "Learning from imbalanced data in surveillance of nosocomial infection," *Artificial Intelligence in Medicine*, vol. 37, no. 1, pp. 7–18, 2006.

[2] P. K. Chan, W. Fan, A. L. Prodromidis, and S. J. Stolfo, "Distributed data mining in credit card fraud detection," *IEEE Intelligent Systems and Their Applications*, vol. 14, no. 6, pp. 67–74, 1999.

[3] Z.-B. Zhu and Z.-H. Song, "Fault diagnosis based on imbalance modified kernel Fisher discriminant analysis," *Chemical Engineering Research and Design*, vol. 88, no. 8, pp. 936–951, 2010.

[4] Z. Zheng, X. Wu, and R. Srihari, "Feature selection for text categorization on imbalanced data," *SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 80–89, 2004.

[5] N. García-Pedrajas, J. Pérez-Rodríguez, M. García-Pedrajas, D. Ortiz-Boyer, and C. Fyfe, "Class imbalance methods for translation initiation site recognition in DNA sequences," *Knowledge-Based Systems*, vol. 25, no. 1, pp. 22–34, 2012.

[6] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[7] V. López, A. Fernández, J. G. Moreno-Torres, and F. Herrera, "Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6585–6608, 2012.

[8] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 42, no. 4, pp. 463–484, 2012.

[9] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.

[10] R. Ranawana and V. Palade, "Optimized precision—a new measure for classifier performance evaluation," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 2254–2261, July 2006.

[11] T. K. Jo and N. Japkowicz, "Class imbalances versus small disjuncts," *SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 40–49, 2004.

[12] R. C. Prati, G. E. A. P. A. Batista, and M. C. Monard, "Learning with class skews and small disjuncts," in *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence*, pp. 296–306, 2004.

[13] R. Barandela, R. M. Valdovinos, J. S. Snchez, and F. J. Ferri, "The imbalance training sample problem:under or over sampling," in *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 806–814, Springer, 2004.

[14] A. Estabrooks, T. Jo, and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," *Computational Intelligence*, vol. 20, no. 1, pp. 18–36, 2004.

[15] A. Estabrooks, *A combination scheme for inductive learning from imbalanced data sets [M.S. thesis]*, Faculty of Computer Science, Dalhousie University, Nova Scotia, Canada, 2000.

[16] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[17] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," in *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*, pp. 935–942, June 2007.

[18] V. García, J. S. Sánchez, and R. A. Mollineda, "On the effectiveness of preprocessing methods when dealing with different levels of class imbalance," *Knowledge-Based Systems*, vol. 25, no. 1, pp. 13–21, 2012.

[19] R. Barandela, J. S. Sánchez, V. García, and E. Rangel, "Strategies for learning in class imbalance problems," *Pattern Recognition*, vol. 36, no. 3, pp. 849–851, 2003.

[20] G. Wu and E. Y. Chang, "KBA: kernel boundary alignment considering imbalanced data distribution," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 786–795, 2005.

[21] C. Elkan, "The foundations of cost-sensitive learning," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pp. 973–978, Seattle, Wash, USA, 2001.

[22] K. M. Ting, "An instance-weighting method to induce cost-sensitive trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 3, pp. 659–665, 2002.

[23] B. Zadrozny, J. Langford, and N. Abe, "Cost-sensitive learning by cost-proportionate example weighting," in *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM '03)*, pp. 435–442, Melbourne, Fla, USA, November 2003.

[24] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2006.

[25] W. Jing, D. Meng, C. Qiao, and Z. Peng, "Eliminating vertical stripe defects on silicon steel surface by L1/2 regularization," *Mathematical Problems in Engineering*, vol. 2011, Article ID 854674, 13 pages, 2011.

[26] C. L. Castro and A. de Pádua Braga, "Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 888–899, 2013.

[27] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: a hybrid approach to alleviating class imbalance," *IEEE Transactions on Systems, Man, and Cybernetics A*, vol. 40, no. 1, pp. 185–197, 2010.

[28] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 39, no. 2, pp. 539–550, 2009.

[29] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "Adacost: misclassification cost-sensitive boosting," in *Proceedings of the 6th*

Internatinal Conference on Machine Learning, pp. 97–105, Tahoe City, Calif, USA, 1999.

[30] K. M. Ting, "A comparative study of cost-sensitive boosting algorithms," in *Proceedings of the 17th International Conference on Machine Learning*, pp. 983–990, San Francisco, Calif, USA, 2000.

[31] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007.

[32] H. Masnadi-Shirazi and N. Vasconcelos, "Cost-sensitive boosting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 2, pp. 294–309, 2011.

[33] X. Wang, S. Matwin, N. Japkowicz, and X. Liu, "Cost-sensitive boosting algorithms for imbalancedmulti-instance datasets," *Advances in Artificial Intelligence*, vol. 7884, pp. 174–186, 2013.

[34] Y. Zhang and D. P. Wang, "A cost-sensitive ensemble method for class-imbalanced datasets," *Abstract and Applied Analysis*, vol. 2013, Article ID 196256, 6 pages, 2013.

[35] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, part 2, pp. 119–139, 1997.

[36] X. Wu, V. Kumar, Q. J. Ross et al., "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[37] R. Caruana and A. Niculescu-Mizil, "Data mining in metric space: an empirical analysis of supervised learning performance criteria," in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04)*, pp. 69–78, Seattle, Wash, USA, August 2004.

[38] R. Alaiz-Rodriguez, N. Japkowicz, and P. Tischer, "A visualization-based exploratory techniquefor classifier comparison with respect to multiple metrics and multiple domains," in *Proceedings of the 15th European Conference on Machine Learning*, pp. 660–665, Antwerp, Belgium, 2008.

[39] Y.-F. Li, J. T. Kwok, and Z.-H. Zhou, "Cost-sensitive semi-supervised support vector machine," in *Proceedings of the 24th AAAI Conference on Artificial Intelligence and the 22nd Innovative Applications of Artificial Intelligence Conference (IAAI '10)*, pp. 500–505, July 2010.

[40] V. García, J. S. Sánchez, and R. A. Mollineda, "Theoretical analysis of a performance measure for imbalanced data," in *Proceedings of the 20th International Conference on Pattern Recognition (ICPR '10)*, pp. 617–620, Istanbul, Turkey, August 2010.