Noname manuscript No.
(will be inserted by the editor)

A Cost-sensitive Algorithm with Local and Global Consistency

Weifeng Sun · **Jianli Sun**

Received: date / Accepted: date

Abstract Assuming that misclassification costs among different categories are equal, traditional graph based semi-supervised classification algorithms pursuit high classification accuracy. However, in many practical problems, misclassifying one category as another always leads to higher (or lower) cost than that in turn, such that, higher classification accuracy generally not means lower cost, which is more serious in these problems. Cost-sensitive classification methods enable classifiers to pay more attention to data samples with higher cost, and then attempt to get lower cost by ensuring higher classification accuracy of the category with higher cost. In this paper, we bring cost sensitivity to local and global consistency (LGC) classifiers, and propose the CS-LGC (cost-sensitive LGC) methods, which can make better use of semi-supervised classification algorithms, and ensure high classification accuracy on the basis of reducing overall cost. At the same time, since the improved algorithm may bring some problems due to unbalanced data, we introduce the SMOTE algorithm for further optimization. Experimental results of bank loan and diagnosis problems verify the effectiveness of CS-LGC.

Keywords Graph based semi-supervised classification · Cost-sensitive · CS-LGC and CSS-LGC · SMOTE

Supported by the Natural Science Foundation of China under Grant No. 61471083, 61300017, the Fundamental Research Funds for Central Universities (DUT15QY28)

W. Sun
School of Software, Dalian University of Technology
DaLian, 116620 - China
E-mail: wfsun@dlut.edu.cn

J. Sun
School of Software, Dalian University of Technology
DaLian, 116620 - China
E-mail: sjl_dlut@163.com

1 Introduction

Recently, machine learning classification algorithms have achieved sound development in multimedia identification, health care, finance, and many other applications. Among them, Graph-based Semi-Supervised Classification (GSSC)[1] based on perfect graph theory, which is relatively straightforward and easy to understand, has become one of the hottest research focuses. This method can make better use of the information from labeled data and a great deal of data distribution information mined from unlabeled data, then solves the problem of the less labeled data and the huge labeling cost.

High classification accuracy is the target of GSSC algorithm, which assumes that the cost of misclassification among different categories is the same. However, in many practical problems, the costs of category classifications are different, especially in the fields of medical service, finance and network security and so on.

For example, in the bank loan problem, banks decide whether to approve the loan based on the loan applicants' personal information, which can be viewed as a binary classification problem. In the process of approving a bank loan, the cost of approving an applicant who is unable to repay the loan is much higher than that of approving an applicant who can repay the loan. Similarly, in the disease diagnosis, the cost of misdiagnose unhealthy patients as healthy is much higher than that of misdiagnose healthy patients as unhealthy. When the cost differences are inconsistent among different categories, a classifier paying more attention to the accuracy of higher cost classification problem is more practical than those that treat all categories equally.

Cost-sensitive learning methods [2–6] takes the inconsistencies of cost among categories into account to lower the overall cost for the target. This method defines the static cost matrix to make classifiers pay more concern to the sample data with higher cost, and improves classification accuracy of higher-cost category. In the semi-supervised classification problems, labels account for a small portion of the data and most data are unlabeled. Actually, the traditional semi-supervised classification treats all categories equally, which cannot guarantee a lower overall cost. Meanwhile, cost-sensitive learning may arise less fit due to limited data set of tags, which could lead to lower classifier accuracy and weaker generalization ability. Besides, in the semi-supervised learning, we often encounter the situation called uneven data problem, in which different types of samples have different numbers of labeled data.

In summary, researches on cost-sensitive semi-supervised classification algorithms for unbalanced datasets are of great significance to the development of finance, medical fields, network security and many other areas.

2 Related Work

The effectiveness of semi-supervised learning algorithm depends on the three assumptions: manifold hypothesis[7,8], clustering hypothesis[**?**] and smoothness hypothesis. GSSC based on manifold hypothesis builds a graph to describe the data, as well as the relationship between the data. In this graph, nodes represent the data samples while edges with weight represent the relationships between samples. At the same time, the larger the weight is, the higher the similarity of the samples have. The process that GSSC classifiers assign labels to unlabeled data is the process of label propagation in the figure. Label Propagation algorithm Budyytis et al. proposed in [9] can calculate the probability of transfer among tag data by the topology and the similarity between the samples in graph, and make a label transfer by combining node out-degree. A method of Gaussian fields and harmonic functions proposed by Zhu et al. in [10], which makes discrete prediction function slack to become continuous prediction function, considers the transfer probability samples fully and have a label transfer in *k*-connection diagram. Local and global consistency, LGC proposed by Zhou et al. in the [11] introduced clustering hypothesis and had a label transfer by using local and global consistencies. LGC algorithm gives a rigorous mathematical logic derivation and proves the convergence. However, these related work never considers the problem of inconsistent classification costs.

Cost-sensitive learning method is an effective way to solve the problem of the inconsistency of costs. Cost-sensitive learning method introduces cost matrix to describe the inconsistency of costs among categories and get global minimum cost. The cost-sensitive classification methods can be divided into two categories: Rescale and Reweight, depending on the representations of cost. In the method of Rescale, differences in cost are described as differences among the numbers of samples, such as Cost-sensitive sampling[2], Rebalance[3] and Rescale new[4] etc. This method constructs different sample data sets according to the difference in costs which make classifiers' decision face prefer samples with more costly category. Reweight method describes differences in costs by differences in weight among samples of different types. In the method of Reweight, samples with costly category have a higher weight, which make a greater impact to classifier. MetaCost[5] is a typical representative of such Reweight methods. MetaCost based on Bayesian risk theory adds the cost of the sensitive nature for Non-cost-sensitive classification algorithm by using bagging[12].

AdaBoost algorithm was proposed in [13] which offers the possibility for multiple weak classifiers aggregating into a global strong classifier. AdaCost method[6] was proposed as a cost-sensitive classification algorithm based on AdaBoost. AdaCost is based on the Reweight at the same time, and introduces cost performance function for classifiers by heuristic strategy. AdaCost forces classifier to pay more attention to costly samples, hence it shows some advantages in cost-sensitive classification problems. However, cost performance function introduced in the theoretical analysis has not been verified and damages the most important characteristics of Boosting, which makes the algorithm not converge to the Bayesian decision. Qin etc. in [14] did try to combine semi-supervised classification algorithms with cost-sensitive learning methods, and improved classical EM algorithm by introducing misclassification cost in the process of probability assessment.

In this paper, advantages of many unlabeled data are fully taken. We use the method of Rescale to describe cost inconsistency and introduce cost-sensitive nature for LGC algorithm classic semi-supervised classification algorithm. We propose a cost-sensitive LGC method, CS-LGC algorithm. At the same time, we take the impact caused by unbalanced data into account for CS-LGC algorithm, improve the CS-LGC algorithm by proposing the average similarity concept and introduce SMOTE algorithm. The main contributions of this paper are as follows:

(1) Advantages of mass unlabeled data are fully taken in CS-LGC algorithm to improve the classification accuracy and deduce the global cost.
(2) CS-LGC algorithm, which is based on AdaBoost method, aggregates weaker classifiers as a strong classifier and prevents AdaBoost method from destroying the nature of Boosting.
(3) Propose CSS-LGC algorithm. We take the impact of unstable classification accuracy caused by unbalanced data into account for CS-LGC algorithm, and we propose optimized CS-LGC algorithm based on CSS-LGC algorithm.
(4) Demonstrate the effectiveness of the CS-LGC algorithm with its optimized algorithm, CSS-LGC algorithm, by experiment on German Credit Data Set, and verify the rationality of them.

3 CSS-LGC Algorithm

3.1 Problem Definition

Many problems can be described as binary classification problems (multiple class classification problems can be split into multiple binary classification problem), so this paper simply discusses binary classification problem. The two classes can be defined as positive class whose sample number is denoted as N_+, and negative class with the sample number denoted as N_-. We further assume C_{-+}, the cost that a sample in negative class is classified into positive class is much bigger than that in turn. $\text{X} = \{x_1, x_2, \ldots, x_l, x_{l+1}, \ldots, x_{l+u}\}$ is a set of data samples, where l is the number of labeled data, and $\{x_1, x_2, \ldots, x_l\}$ forms the initial labeled data sets. u represents the number of unlabeled data, thus $U = l + u$ denotes the total number of samples. x_i is a multidimensional vector of the i-th data with several features, which can be viewed as a point in higher dimensional space. Those data points and their relationships can be described by a fully connected graph $G = (V, E)$, where $V \in R^{N \times P}$ represents the vertex set of G, P is the feature value of each data sample. we have $V = X$ here, and E is the edge set of G. Similarity among data points is important in classification, clustering and other machine learning fields. In LGC algorithm, similarity among data points is described by weight matrix $W \in R^{N \times N}$. Labeled data information is described by labeled information matrix $Y \in L^{N \times 2}$, where $L \in \{+1, -1\}$. If data i belongs to category j, then $Y_{ij} = 1$, otherwise $Y_{ij} = 0$. The initial labeled information matrix only has a little labeled information.

Liu and Zhou mentioned that the cost of misclassification can be standardized in [15], which will not affect the best decision when calculation is simplified. In this paper, we let $c_{+-} = 1$, so c_{-+} is a real number larger than 1. In order to make semi-supervised classifier cost sensitive, We introduce sample cost information for the initial label matrix, denote cost difference using different initial labeled information volumes, and process the initial label matrix as Formula 1 shows:

$$YN_{.i} = T_0 \circ Y_{.i} \tag{1}$$

$N_{.i}$ indicates the i-th column of the processed initial label matrix. $\circ$ indicates Adama product. T_0 indicates the cost of sample data at the initial time. In addition

$$T_0(x_i) = \begin{cases} 1/N_+ & \text{if } x_i \text{ is positive class} \\ C_{-+}/N_- & \text{if } x_i \text{ is negative class} \end{cases} \tag{2}$$

CS-LGC method is designed to classify all data samples using the information of graph G with unknown Y and T_0. Our goal is to ensure unlabeled data to obtain class label as many as possible without changing the information of labeled data, and at the same time, minimize the global classification cost without damaging classification accuracy.

3.2 Algorithm Process

CS-LGC method is a boosting process that trains semi-supervised classifiers in each iteration and update labeled data set based on the performance of the classifiers. At the initial time, semi-supervised classifier h_0 is trained using traditional LGC algorithm according to preprocessed initial label matrix, and we further calculate error rate by Formula 3.

$$\varepsilon_0 = \frac{\sum_{i=1}^{N} I\left(y_{i,h_0(x_i)} = 0\right)}{N} \tag{3}$$

Here I is a binary function, whose value is 1 if conditions are satisfied, or is 0 else. $h_0(x_i)$ shows classification results for x_i at the initial time. Then according to the AdaBoost algorithm, cost information of misclassified samples increases while cost information of correctly classified samples decreases following Formula 4,

$$T_{t+1}(x_i) = \frac{T_t(x_i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_{i,h_t(x_i)} = 1 \\ e^{\alpha_t} & \text{if } y_{i,h_t(x_i)} = 0 \end{cases} \tag{4}$$

where T_{t+1} represents the updated cost matrix, and

$$Z_t = \sum_{i=1}^{N} T_t(x_i) \times \begin{cases} e^{-\alpha_t} & \text{if } y_{i,h_t(x_i)} = 1 \\ e^{\alpha_t} & \text{if } y_{i,h_t(x_i)} = 0 \end{cases} \tag{5}$$

where α_t represents the weight of classifier h_t in the final global classification at time t, and can be calculated by

$$\alpha_t = \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \tag{6}$$

The main objective of cost-sensitive classifier is to minimize global classification. To avoid the impact of data set scale on the results, the calculation of global cost in our model use the mean cost method brought by Seiffert et al[16]. Since we have to standardize classification cost of our model, the corresponding mean cost needs to be updated according to Formula 7:

$$PEC = \frac{\#f_{pos} + \#f_{neg} \cdot C_{-+}}{\#t_{pos} + \#t_{neg} + \#f_{pos} + \#f_{neg}} \tag{7}$$

where PEC denotes the average cost of classifiers, $\#t_{pos}$ and $\#f_{pos}$ denote the number of positive class samples classified correctly and classified incorrectly respectively. $\#t_{neg}$ and $\#f_{neg}$ represent the corresponding information of negative class. The global cost we refer to here that is indeed the average cost. From Formula 7 we can see that the increment of correct classified samples does not increase the global cost.

Next, we update the label matrix YN according to Rescale method. Specifically, when the cost of x_i increases, means there is a wrong classification. Then a certain number of data that have the highest similarity with x_i are chosen from unlabeled data set and added into labeled data set. The class and cost information of the newly added data are the same with x_i. On the contrary, when sample data x_i's cost decreases, and the decreasing range is larger than the threshold ts, then we will remove x_i and its label information from the labeled data set.

After then, we train the classifier of the next time according to updated label matrix and repeat the process until the global cost difference between two training time step is less than a certain threshold. Finally, a global classifier can be obtained by the way of weighted voting.

$$Y_{final}(x_i) = sign\left(\sum_{j=1}^{M} \alpha_j h_j(x_i)\right) \tag{8}$$

Here M denote the number of classifier when the iteration ends.

In summary, the process of CS-LGC method is shown in algorithm 1:

Algorithm 1 CS-LGC

```
Input: data set $X$, the initial label matrix $Y$
1: add cost information for the initial label matrix according to Formula 1
2: while $\Delta PEC \geq threshold$ do
3:     train LGC semi-supervised classifier
4:     update the cost matrix by Formal 4 according to the result
5:     update labeled data set applying Rescale method
6:     update the initial label matrix according to Formula 1
7: end while
Output: $Y_{final}(x_i) = sign(\sum_{j=1}^{M} \alpha_j h_j(x_i))$ , $PEC = \frac{\#f_{pos}+\#f_{neg}\cdot C_{-+}}{\#t_{pos}+\#t_{neg}+\#f_{pos}+\#f_{neg}}$
```

4 Improved CS-LGC algorithm: CSS-LGC

4.1 Defect Analysis of CS-LGC algorithm

The cost matrix will be updated every time after classifiers finishing training, which increases the corresponding cost information of incorrect classifications and that of correct classifications, forcing classifiers to pay more attention to the classification accuracy of high-cost samples. As mentioned above, the specific techniques to update labeled data set can be referred as: when x_i of sample data increases, which suggests the classification errors of classifiers, our measure is to select the data which has the highest similarity with x_i depend on the change range of x_i, and add it into the data set. It may occur at this time that we can only select out those having the highest similarity with x_i from unlabeled data and labeled it as the same class with x_i without the judgment of the possibler class which the selected unlabeled data is supposed to belong to. If the selected unlabeled data has a similarity closer to another class or simply the opposite class of x_i' class and then we mark it as the same class as x_i, it will lead to error accumulation in the following training and influence the performance of classifiers.

In order to solve the problems above, we propose a new method. After updating cost matrix, the specific measures of this method is to apply Rescale to updating label matrix as follows:

Step 1 For each remaining unlabeled data, we calculate its average similarity with all data marked as positive class. Then we obtain the average similarity of each unlabeled with all samples marked as positive.

Step 2 Calculate the average similarities of all unlabeled data with all samples marked as negative class, Then we obtain the average similarity of each unlabeled with all samples marked as negative.

Step 3 The selection of unlabeled data is supposed to meet two conditions: 1) The average similarity of the unlabeled data ought to be closer to the class to mark; 2) Among the unlabeled data with average similarity closer to that class, those selected should be picked out in descending order by average similarity.

However, one situation may be faced with above: According to the updated cost matrix, Q data need selecting from the labeled data to join the negative class. However, it turns out that the number of the unlabeled data is smaller than Q after calculating the average similarly. The average similarity is more close to either positive class or negative class. That is the problem we often encounter with unbalanced data sets, at this time this situation may come: classification precision appears fluctuant, which leads to unstable performance.

In response to this problem of unbalanced data sets, we introduce in SMOTE algorithm, i.e. using the method of synthetic artificial virtual data to solve the serious problem of unbalanced data sets, so as to solve the problem of unstable performance of the classifiers.

4.2 CSS-LGC algorithm Process

SMOTE algorithm generates virtual data through a synthetic way. For each minority category sample, the method looks for its k samples with the highest similarity. Then Upward sampling method is applied. If the amount of upward sampling samples is set as N, it randomly selects N samples from all k samples, which can be represented as x_i $(i = 1, 2, \ldots, N)$, then conducts the linear interpolation between minority class x and x_i. Next, it constructs new minority samples $p_1, p_2, ..., p_N$ and

$$p_i = x + rand(0,1) * (x_i - x), \quad i = 1, 2, \ldots, N \tag{9}$$

The process of CSS-LGC algorithms is roughly same with CS-LGC, except the difference when CSS-LGC algorithm uses Rescale to update labeled data sets. In the CSS-LGC algorithm, after updating the cost matrix, Rescale is applied to update label matrix, then, it adds SMOTE method and optimizes the overall process of updating the label matrix. The specific implementation process is as follows:

1) According to the change of cost matrix before and after the update, we determine the amount of required unlabeled data chosen in each category.

Algorithm 2 CCS-LGC

```
Input: data set $X$, the initial label matrix $Y$
1: add cost information for the initial label matrix according to Formula 9
2: while $\Delta PEC \geq threshold$ do
3:     train LGC semi-supervised classifier
4:     update the cost matrix by Formal 4 according to the result
5:     update labeled data set applying Rescale method
6:     update the initial label matrix according to Formula 1
7: end while
Output: $\mathrm{Y}_{final}(x_i) = sign(\sum_{j=1}^{M} \alpha_j h_j(x_i))$ , $PEC = \frac{\#f_{pos}+\#f_{neg}\cdot C_{-+}}{\#t_{pos}+\#t_{neg}+\#f_{pos}+\#f_{neg}}$
```

2) First of all, we get average similarity respectively for samples that have been marked as positive class from the remaining unlabeled data. Then we get the average similarity between the samples of each unlabeled and all samples that have been marked as positive class, denote them as $p_1, p_2, \ldots, p_u$, where u represents the number of unlabeled samples. In the same way, we use all unlabeled data to get an average similarity for the samples that has been marked as negative class, which denoted as $p'_1, p'_2, \ldots, p'_u$. Here $p_1, p_2, \ldots, p_u$ and $p'_1, p'_2, \ldots, p'_u$ are respectively calculated by Formula 10 and Formula 11,

$$p_j = (\sum_{i=1, \mathrm{Y}_{i1} = +1}^{l} w_{(l+j)i})/N_+, \quad j = 1, 2, \ldots, u. \tag{10}$$

$$p'_j = (\sum_{i=1, \mathrm{Y}_{i1} = -1}^{l} w_{(l+j)i})/N_-, \quad j = 1, 2, \ldots, u. \tag{11}$$

where N_+ and N_-, respectively denote the number of the positive and negative samples in the data that have been marked, Y is the label matrix, and W is the weight matrix.

3) According to select unlabeled data, two conditions need meeting: first, the average similarity of unlabeled data must be closer to the class which will be marked as; Second, data with average similarity closer to this category unlabeled data should be chosen in accordance with descending order of average similarity and the number of unlabeled data to select calculated in the first step. At this time, it will have two cases:
   a) The number of average similarity of unlabeled data closer to this class which will be marked as is enough, that is to say, we can find out enough samples from unlabeled data for this class.
   b) There are few samples with average similarity closer to the class in the unlabeled data, namely, the number of unlabeled data with similarity to be marked in this class is littler than the other class, or only a small portion of data are more similar to this class.

   If this is the first case, after completing marking the selected unlabeled data, we can simply exit the Rescale process, then update the initial label matrix in line with formula 1 and go on with the following learning processes. Otherwise, as the second case occurs, we jump to the fourth step to process.

4) After making some artificial virtual data using SMOTE method, our method jumps to the second step, and continue conduct, until the number of artificially

manufactured unlabeled data is more closer to the number that this class need to be marked, and when the data simples is abundant, jump to the process of the first case in step 3.

In the process of above, when synthesizing artificially data by using SMOTE algorithm for each minority class that need unlabeled data, instead of applying the method of upward sampling, our solution is to determine the number of unlabeled data which are to be selected according to the cost ratio of before to after updating. For instance, when the cost ratio is $r(r > 1)$, then the number of unlabeled data to select is $r - 1$. At this time, we need to select $r - 1$ samples from the k unlabeled samples with the highest similarity and do linear interpolation with minority class samples. The Fig 1 demonstrates the process of updating label matrix using the combination of Rescale method and SMOTE method in detail.

Through improving Rescale process of the CS-LGC algorithm, we solve the defection of classification errors, which may accumulate in the unbalanced dataset and enhance the stability of classifiers' performance.

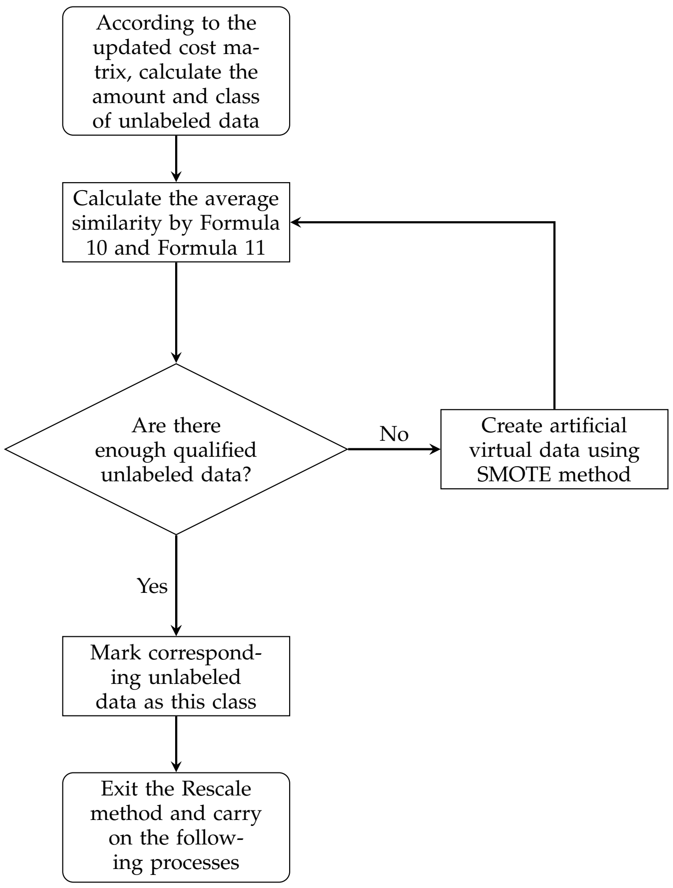
According to the updated cost matrix, calculate the amount and class of unlabeled data
Calculate the average similarity by Formula 10 and Formula 11
Are there enough qualified unlabeled data?
No
Create artificial virtual data using SMOTE method
Yes
Mark corresponding unlabeled data as this class
Exit the Rescale method and carry on the following processes

Fig. 1 The flowchart of updating label matrix using Rescale method in CSS-LGC algorithm

4.3 Performance Analysis of CSS-LGC algorithm

Generally speaking, CSS-LGC algorithm is an improvement based on CS-LGC algorithm, and its core is still CS-LGC algorithm yet, except some methods for unlabeled data. CSS-LGC algorithm has almost the same convergence and time complexity with LGC algorithm.

For convergence of the algorithm, Zhou gives the evaluation function of the semi-supervised classification algorithm based on graph in [11]. According to extremum condition acquired by evaluation function as well as recursion formula of LGC algorithm, Zhou deduces the convergence formula (Formula 12) of LGC algorithm, then proves the convergence of LGC algorithm.

$$F = (I - \beta S)^{-1} Y \tag{12}$$

Here F is the final classification function, $\beta \in (0,1)$, S is a symmetric matrix, and

$$\mathrm{S} = D^{-1/2} W D^{-1/2} \tag{13}$$

where D is a diagonal matrix, whose diagonal elements is the sums of elements of corresponding row vectors in the weight matrix W.

In terms of time complexity, the increased time complexity by introducing cost sensitive feature to semi-supervised classification algorithm based on graph is mainly reflected in the process of boosting and updating cost matrix.the aim of updating the cost matrix is to update the labeled data set, including adding new labeled data and deleting existing labeled data. The adding operation has a time complexity as $O(l \cdot n \lg n) < O(n^2)$, where l is the number of labeled data, while deleting operation has the time complexity of $O(n \lg n)$. The time complexity of boosting process is $O(M \cdot n^3)$, the time complexity of base-classifier LGC algorithm reaching $O(n^3)$ included and M represents the number of base-classifiers. In CSS-LGC algorithm, we mainly improve on the process of updating cost matrix, reflected in SMOTE method. The time complexity of SMOTE method can be expressed as $O(n \cdot p \cdot m)$, where n is the total number of samples, p represents the number of features in each data sample, and m is defined as the ratio of the number of low-cost samples to that of high-cost ones. With a large data volume, p and m can be considered as constants, so the time complexity of SMOTE can be written as $O(n)$. In other words, even if the samples have a large amount of features and a big ratio of sample number of two classes above, SMOTE method can still keep a time complexity no more than $O(n^3)$. In summary, the time complexity of CSS-LGC algorithm is $O(n^3)$.

5 Experimental Results and Analysis

5.1 Data Introduction and Preprocessing

We choose bank loans and disease diagnosis, whose misclassification costs are inconsistent, from professional database UCI[17], to test our CS-LGC and CSS-LGC algorithm. Since the fact that CS-LGC is the improvement of LGC, we will verify whether CS-LGCs global cost will be reduced and whether its accuracy will be lowered compared with LGC. In addition, CS-LGC is a cost-sensitive classification method, so we will compare it with a traditional cost-sensitive algorithm, AdaCost algorithm, to see the performance difference between two algorithms.

Cost-sensitive learning focuses more on sample data of high cost, so in this paper, we implement three experiments to verity the properties of CS-LGC and CSS-LGC: (1) the classification accuracy of high cost category changes with the labeled data; (2) the global classification accuracy changes with the labeled data; (3) the global cost changes with labeled data.

German Credit Data Set[18], which describes the information of bank loan applicants and whether the information of the relevant loan application is approved, was provided by Professor Hans Hofmann Et al. The data set collected 1000 loan applicants' information, including current deposit, borrowing time, fixed asset, working condition as well as whether they have guarantee and so on. There are 20 characteristic attributes in all. Finally, the applications of 700 person were approved while the other 300 applications weren't. We define the people who got the approvals as positive class, and those who did not as the negative class, then set $C_{+-} = 1$, $C_{-+} = 5$, and initialize the cost of CS-LGC after standardizing process. Different features will represent in different value ranges, such as applicants' genders only have 0 and 1, two small value, while borrowing time could reach 48 (months) or even much larger. To eliminate this difference, feature values of characteristics for the applicants should be linear normalized as Formula 14.

$$x_{ij}' = \frac{x_{ij} - x_{i\min}}{x_{i\max} - x_{i\min}} \tag{14}$$

x_{ij}' denotes the results of the j-th attribute value of the i-th applicant after normalization, $x_{i\max}$ and $x_{i\min}$ respectively denote the maximum and minimum value in all attribute values of applicant i.

5.2 Threshold Discussion in CSS-LGC

Since we are to verify the advantages of CSS-LGC algorithm on German Credit Data Set, the experiments to analyze the choice of two critical parameters' thresholds in CSS-LGC are implemented on this data set.

These two thresholds need to be set in CS-LGC and CSS-LGC algorithm: The first one is ts, the decent degree threshold of sample data x_i's cost. When the cost of x_i decreases more than ts, it will be deleted from the labeled data set. Another is *threshold*, the global cost difference threshold of classifier between two iterations, that is to say, When the global cost difference between two consecutive iterations is less than the *threshold* value, the classifier will stop iteration. Here, we only discuss about the *threshold* of CSS-LGC algorithm.

threshold is set as 10% of the initial global cost. We verify the selection of the two thresholds of CSS-LGC using German Credit Data Set.

5.2.1 ts

According to Fig. 2, Fig. 3 and Fig. 4, we select 50% data from this data set as labeled data for that when the percentage of labeled data reaches 50%, the global classification accuracy and global cost can hardly change, which is benefit to analyzing the choice of threshold ts.

From Table 1 we can see that the negative classification accuracy and the global classification accuracy both continuously increase when *ts* increases. Because with the percentage of labeled data fixed and *ts* set as a little number, i.e. the cost of the sample data decreases just a little, the sample gets rid of from labeled data, which reduces the amount of labeled data, the negative classification accuracy and the global classification. When *ts* grows over 50%, the negative classification accuracy and the global classification accuracy of CSS-LGC show little change and almost stable. It means that samples with over 50% range of take up just a little portion in the data set, where the choice of *ts* make little difference on the accuracy of classification. Therefore, the value of *ts* should be set as a number greater than or equal 50%.

It is also shown in Table 1 that the global cost decreases progressively overall with *ts* increasing. Because the continuously increased *ts* leads to the deducted cost sensitivity of CSS-LGC algorithm, which diminishes the attention to high-cost class samples and deducts the global cost of classifiers. At the same time, from this table we can see when *ts* grows up to 50%, the global cost begins to be stable.

Through a comprehensively analysis of the impact on the negative classification accuracy, global classification accuracy and global cost of CSS-LGC algorithm with *ts*, we consider 50% as a suitable value of *ts* in our experiments.

ts	Negative classification accuracy	Global classification accuracy	Global cost
0.05	0.546	0.7985	0.82205
0.1	0.6876	0.8035	0.60615
0.15	0.7946	0.8274	0.42896
0.2	0.8546	0.8293	0.33755
0.25	0.8937	0.8305	0.27675
0.3	0.9254	0.8327	0.22906
0.35	0.9348	0.8346	0.21394
0.4	0.9365	0.8348	0.21102
0.45	0.9367	0.8349	0.21052
0.5	0.946	0.835	0.1965
0.55	0.9463	0.8352	0.19591
0.6	0.9464	0.8352	0.19576
0.65	0.9465	0.8353	0.19554
0.7	0.9465	0.8353	0.19554
0.75	0.9465	0.8353	0.19554
0.8	0.9465	0.8353	0.19554
0.85	0.9465	0.8353	0.19554
0.9	0.9465	0.8353	0.19554
0.95	0.9465	0.8353	0.19554
1	0.9465	0.8353	0.19554

Table 1 Impact and Analysis Table with *ts*

5.2.2 threshold

To observe the impact that *threshold* has on the results of classifiers, we create a change curve graph to describe the tendency of negative class classification accuracy, global classification accuracy and global cost(average cost) in CSS-LGC algorithm when *threshold* increases from 0% to 100%. We still select Matlab R2010a as experimental platform and 50% data as labeled.

As it shown in Table 2, when *threshold* changes from 6 to 15, as the iteration end condition of classifiers-the difference of global costs between the last two iterations is littler than 6% to 10% of the global cost at the initial time, the negative class classification accuracy of CSS-LGC appears continuously declining with a small range. The global classification accuracy of CSS-LGC continues to shrink with the change curve of *threshold*, mainly because with the increase of *threshold*, the iteration end condition become looser gradually, which may contributes to the fewer training times of base-classifiers and the lower accuracy of classifiers further. When *threshold* grows form 6 to 10, where changes appear slow and the global cost turns relatively little. While *threshold* changes from 11 to 15, *PEC* changes a little more steeply, meanwhile, the *PEC* at that time has a significant growth compared to that when *threshold* set as 6 to 9.

We set the value of *threshold* as 10 in experiments. Two reasons support the setting: on the one hand, when it is 10, precision and the global cost are still within an acceptable range; On the other hand, although the corresponding results of threshold set as 6,7,8,9 are better than 10, they take much more time and converges slower. To sum up, we consider 10 an optimal value for *threshold*.

threshold	Negative class classification accuracy	Global classification accuracy	Global cost
6	0.9546	0.85	0.1731
7	0.95186	0.845	0.18071
8	0.9486	0.8425	0.18735
9	0.947	0.839	0.1922
10	0.942	0.835	0.2025
11	0.935	0.83	0.2165
12	0.923	0.82752	0.23624
13	0.908	0.8247	0.26071
14	0.8967	0.8189	0.28172
15	0.8886	0.8135	0.29765

Table 2 Impact and Analysis Table with *threshold*

5.3 Experiment Results and Analysis

The data set mentioned in 5.1 will contribute to the problems of CS-LGC algorithm in our experiments, which relates to the ratio of samples in the data set to misclassification cost. CSS-LGC algorithm is an improvement of CS-LGC algorithm by solving the problems of error accumulation on unbalanced data set and unstable performance in CS-LGC algorithm. To verify the superiority of CSS-LGC algorithm to CS-LGC algorithm on cost-sensitive learning, three experiments are set on the data set: (1) the classification accuracy of high cost category changes with the labeled data; (2) the global classification accuracy changes with the labeled data; (3) the global cost changes with labeled data.

As it mentioned in 4.2, when synthesizing artificially data by using SMOTE algorithm for each minority class that need unlabeled data, instead of applying the method of upward sampling, our solution is to determine the number of unlabeled data to select according to the cost ratio of before to after updating. For instance, when the cost ratio is $r(r > 1)$, then the number of unlabeled data to select is $r - 1$. At this time, we need to select $r - 1$ samples from the k unlabeled samples with the highest similarity and do linear interpolation with minority class samples.

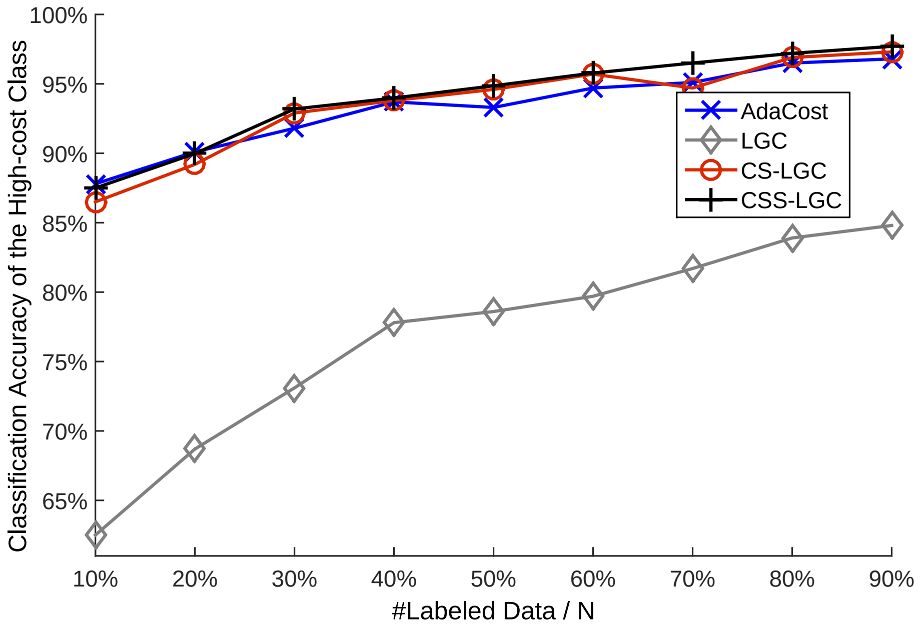

Fig. 2 Changes in Classification Accuracy of High-cost Class with Labeled Data Scale

Fig. 2 describes the changes in classification accuracy of high-cost class. we can see that the classification accuracies of CS-LGC algorithm and AdaCost algorithm always stay at a high level, while the high-cost class classification accuracy of LGC algorithm is lower than that of CSS-LGC algorithm and AdaCost algorithm, because CSS-LGC algorithm and AdaCost algorithm pay more attention to samples with high cost than LGC algorithm, which shows their property of cost-sensitive. CSS-LGC algorithm has a relatively stable performance and little fluctuation on classification accuracy. Besides, the high-cost class classification accuracy of CSS-LGC algorithm is always higher than other three algorithms, due to the introduced conception of average similarity in the process of updating label matrix. In CSS-LGC algorithm, we calculate the average similarities of unlabeled data with all data labeled as either positive or negative class respectively before selecting unlabeled data, and determine whether unlabeled data meet the two conditions to select by comparing the average similarities with both two classes, in case of misclassification and error accumulation in later classification processes. Furthermore, SMOTE method is introduced in CSS-LGC algorithm to solve the problem of local convergence caused by unbalanced data. Therefore, CSS-LGC algorithm has the best overall performance among all four algorithms.

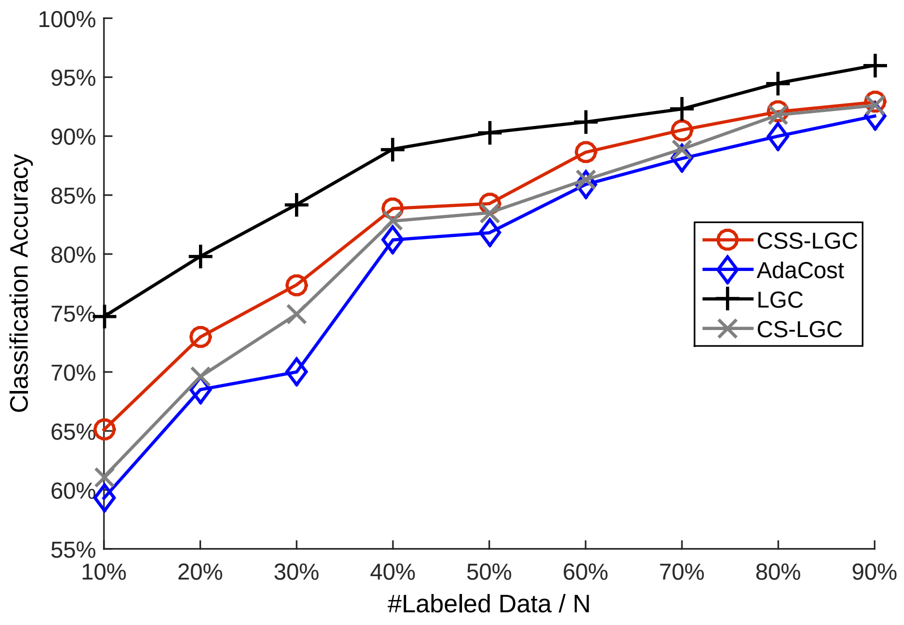

Fig. 3 Changes in Classification Accuracy with Labeled Data Scale

Fig. 3 shows the curve of global classification accuracy with labeled data scale increasing. Similar to above, the average classification accuracy of LGC algorithm is about 90%, 10% higher than that of CS-LGC and AdaCost, and 8% higher than that of CSS-LGC. It indicates that cost-sensitive classification algorithms such as CS-LGC, CSS-LGC and AdaCost pay more attention on high-cost class samples. Because the number of low-cost class samples is larger, the global classification accuracies of those three algorithms are lower than that of LGC. But the classification accuracy can still stay at an acceptable range, and with the increase of labeled data, the global classification accuracy goes up obviously. When the percentage of the labeled data is up to a certain constant, the global classification accuracy of all four algorithms reach a higher level. Though CSS-LGC algorithm has a lower classification accuracy than that of LGC algorithm, which is higher than that of CS-LGC and AdaCost algorithm by about 2% yet in the curves. It soundly implies that the weakness of minority class in the data set begins to appear as unlabeled data marked as labeled constantly because of the usage of SMOTE method in Rescale method. At this time, some artificial data are created by SMOTE method in CSS-LGC algorithm, and participate in comparison loop by calculated similarities, until unlabeled data closer to minority class are synthesized out, which avoids the error accumulation and unstable performance in CS-LGC algorithm and improve the classification accuracy.

Fig. 4 shows the global cost changing with the size of labeled data. In Fig. 4, the global costs of CSS-LGC, CS-LGC and AdaCost are obviously lower than LGC, by about 0.2 on average. We can know from both Fig. 3 and Fig. 4, cost-sensitive classification sacrifices some classification accuracy, in exchange for a lower global cost, and CSS-LGC performs best among the four. The global cost of AdaCost algorithm

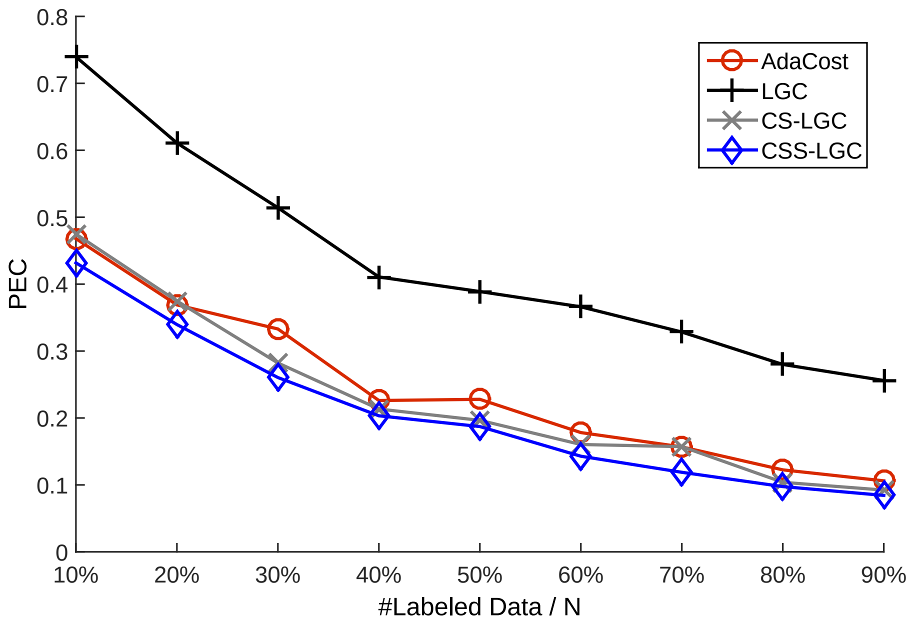

Fig. 4 Changes in PEC with Labeled Data Scale

shows the tendency of increase and great fluctuation when the percentage of labeled data increases as 30% and 50% during training process, which origins from that AdaCost algorithm does not converge at Bayesian decision at that time. CS-LGC algorithm also has a slight tendency of increase on average cost, because Rescale method can only select data to mark from unlabeled data by the value of similarities in CS-LGC algorithm, which eliminates the problem of data insufficiency in minority classes and error accumulation. While CSS-LGC algorithm keeps a decrease trend on global cost and a superior performance to other three algorithms. Besides, CS-LGC algorithm converges faster when the labeled data size increases, and the iteration time decreases, whose average value is between 10 and 20.

From Fig. 2, Fig. 3 and Fig. 4, we can see, in the initial stage of data set training, there is little difference between CSS-LGC and CS-LGC algorithm. That is because in the beginning, the little amount of minority class samples do not work in the data set. At this time, there are enough unlabeled data for minority class can be chosen. But with the proceeding of the training, the number of unlabeled data is declining. Then the weakness of minority class data will be evident for the selection of unlabeled data using CS-LGC algorithm, which does not take the number of minority class samples into account.That will cause error accumulation, and undermine performance of CS-LGC algorithm. However, when the unlabeled data of minority class is trained by CSS-LGC algorithm, SMOTE algorithm introduced in works. And with the method of synthesizing, some virtual unlabeled data of weak minority are created. Then unlabeled data of minority class are continuously generated until conditions are met by comparison, which can be accepted by CSS-LGC algorithm. CSS-LGC algorithm improves the performance and repairs the defect caused by unbalanced data in CS-LGC algorithm. As suggested above, CSS-LGC algorithm presents better overall performance compared to CS-LGC algorithm.

6 Conclusions

In this paper, we proposed CS-LGC algorithm that introduced cost sensitivity into classic semi-supervised classification algorithm based on AdaBoost method, and obtained better performance global classifier in the way of weighted voting for weak classifiers.

We take into account the defects that may be caused by imbalanced data set, improved CS-LGC algorithm and proposed CSS-LGC algorithm. In semi-supervised learning, CSS-LGC algorithm makes full use of a large number of unlabeled data. Considering heterogeneous distribution of misclassification cost in real life, cost sensitivity is introduced into LGC algorithm. In the process of label matrix update, CSS-LGC algorithm improves its comprehensive performance and generalization using SMOTE method that can eliminate the effect of imbalanced data sets. Therefore, CSS-LGC algorithm is practical and instructive in solving real-life problems.

However, CSS-LGC algorithm's performance will be compromised because similarity computations will be done for more times and take more time to find out the required unlabeled data in the Rescale process. Thus in our future work, we plan to simplify the calculation of average similarity, optimize SMOTE method and improve overall performance of CSS-LGC further.

References

1. X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.
2. Y. Gao and J. Wang, "Active learning method of bayesian networks classifier based on cost-sensitive sampling," in *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on*, vol. 3, pp. 233–236, IEEE, 2011.
3. C. Elkan, "The foundations of cost-sensitive learning," in *International joint conference on artificial intelligence*, vol. 17, pp. 973–978, Citeseer, 2001.
4. Z.-H. Zhou and X.-Y. Liu, "On multi-class cost-sensitive learning," *Computational Intelligence*, vol. 26, no. 3, pp. 232–257, 2010.
5. P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 155–164, ACM, 1999.
6. W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "Adacost: misclassification cost-sensitive boosting," in *ICML*, pp. 97–105, 1999.
7. R. Urner, S. Shalev-Shwartz, and S. Ben-David, "Access to unlabeled data can speed up prediction time," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 641–648, 2011.
8. Z.-H. Zhou and M. Li, "Semi-supervised learning by disagreement," *Knowledge and Information Systems*, vol. 24, no. 3, pp. 415–439, 2010.
9. I. Budvytis, V. Badrinarayanan, and R. Cipolla, "Label propagation in complex video sequences using semi-supervised learning.," in *BMVC*, vol. 2257, pp. 2258–2259, Citeseer, 2010.
10. X. Zhu, Z. Ghahramani, J. Lafferty, *et al.*, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, vol. 3, pp. 912–919, 2003.
11. D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in neural information processing systems*, vol. 16, no. 16, pp. 321–328, 2004.
12. T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Comparing boosting and bagging techniques with noisy and imbalanced data," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 41, no. 3, pp. 552–568, 2011.

13. X. Jin, X. Hou, and C.-L. Liu, "Multi-class adaboost with hypothesis margin," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, pp. 65–68, IEEE, 2010.
14. Z. Qin, S. Zhang, L. Liu, and T. Wang, "Cost-sensitive semi-supervised classification using csem," in *Computer and Information Technology, 2008. CIT 2008. 8th IEEE International Conference on*, pp. 131–136, IEEE, 2008.
15. X.-Y. Liu and Z.-H. Zhou, "The influence of class imbalance on cost-sensitive learning: An empirical study," in *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pp. 970–974, IEEE, 2006.
16. C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano, "A comparative study of data sampling and cost sensitive learning," in *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*, pp. 46–52, IEEE, 2008.
17. M. Lichman, "UCI machine learning repository," 2013.
18. H. Hofmann, "German credit data," 2000.