Geometric Structural Ensemble Learning for Imbalanced Problems

Zonghai Zhu, Zhe Wang, Dongdong Li, Yujin Zhu, and Wenli Du

***Abstract*—The classification on imbalanced data sets is a great challenge in machine learning. In this paper, a geometric structural ensemble (GSE) learning framework is proposed to address the issue. It is known that the traditional ensemble methods train and combine a series of basic classifiers according to various weights, which might lack the geometric meaning. Oppositely, the GSE partitions and eliminates redundant majority samples by generating hyper-sphere through the Euclidean metric and learns basic classifiers to enclose the minority samples, which achieves higher efficiency in the training process and seems easier to understand. In detail, the current weak classifier builds boundaries between the majority and the minority samples and removes the former. Then, the remaining samples are used to train the next. When the training process is done, all of the majority samples could be cleaned and the combination of all basic classifiers is obtained. To further improve the generalization, two relaxation techniques are proposed. Theoretically, the computational complexity of GSE could approach $O(nd\log(n_{\min})\log(n_{\mathrm{maj}}))$. The comprehensive experiments validate both the effectiveness and efficiency of GSE.**

***Index Terms*—Basic classifier, ensemble learning, geometric structure, imbalanced problems, machine learning, relaxation techniques.**

I. Introduction

IT IS now quite common to see imbalanced problems in recent years. For instance, many real-world applications, such as medical diagnosis [1], [2] and fault diagnosis [3], are suffering from the problem. In binary classification, the imbalanced problem usually refers to the situation that one class (minority class) has a small amount of samples but the other (majority class) is with much more samples. In general, traditional classifiers, using overall accuracy as the evaluation criterion, are inappropriate to imbalanced problems [4]. When dealing with imbalanced problems, traditional classifiers tend to misclassify minority samples to improve the overall accuracy. For instance, the optimal classification hyperplane of the support vector machine (SVM) [5] may be bias toward the minority samples. However, the minority samples are relatively more important in the imbalanced problems.

To overcome the drawback mentioned above, methods specially for imbalanced problems are proposed and could be categorized into three primary types including the sampling strategy [6]–[8] to rebalance the distribution of data, the cost-sensitive learning [9]–[11] to assign appropriate misclassification cost to minority samples, and the algorithm modification [12], [13] to make the existing algorithms adapt imbalanced problems.

The sampling strategy can be further categorized into three groups including oversampling, undersampling, and hybrids. The oversampling algorithms [14]–[16] generate reasonable minority samples to balance the distribution of data, while the undersampling algorithms [17], [18] remove majority samples to balance the distribution of data. For hybrids methods [19], [20], they perform relatively better through combining oversampling and undersampling. In cost-sensitive learning, researches [21]–[24] show the effectiveness of cost-sensitive learning.

In algorithm modification strategy, the ensemble learning, which tries to improve the performance of single classifier by combining several complementary classifiers, shows its effectiveness in adaptability and generalization. More concretely, the ensemble learning can be further divided into boosting and bagging [25], [26] types. In boosting-based ensembles, many approaches including AdaCost [27], RareBoost [28], and AdaC1, C2, C3 [29] are proposed and show obvious effectiveness to imbalanced problems by modifying the weight update formula in the iteration of AdaBoost [25]. When the sampling strategy is embedded into boosting-based ensemble [30], the weight of minority samples in every iteration is relatively aggravated. This type of approaches mainly includes SMOTEBoost [31] and RUSBoost [32]. SMOTEBoost adds minority samples according to SMOTE in every iteration, while RUSBoost reduces the majority samples randomly. In this way, the data distribution in every iteration is balanced.

Manuscript received May 7, 2018; revised August 20, 2018; accepted October 9, 2018. Date of publication November 9, 2018; date of current version February 25, 2020. This work was supported in part by the Natural Science Foundation of China under Grant 61672227, in part by the National Science Foundation of China for Distinguished Young Scholars under Grant 61725301, and in part by "Shuguang Program" through the Shanghai Education Development Foundation and the Shanghai Municipal Education Commission. This paper was recommended by Associate Editor Y. Jin. *(Corresponding authors: Zhe Wang; Wenli Du.)*

Z. Zhu and Z. Wang are with the Key Laboratory of Advanced Control and Optimization for Chemical Processes, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China, and also with the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China (e-mail: wangzhe@ecust.edu.cn).

D. Li and Y. Zhu are with the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China.

W. Du is with the Key Laboratory of Advanced Control and Optimization for Chemical Processes, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China (e-mail: wldu@ecust.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2018.2877663

Except for the boosting-based methods, the bagging-based methods also show the great power to tackle imbalanced problems. Similar to boosting, it includes overbagging and underbagging [33], [34]. SMOTEBagging uses SMOTE to generate new minority samples, thus balancing the distribution. For underbagging, it mainly includes asymmetric bagging [35], QuasiBagging [36], and roughly balanced bagging [37]. These algorithms select the majority samples whose number is equal to the size of minority class to make the distribution balanced. Furthermore, hybrid ensemble method, combining boosting and bagging, is an ensemble of ensembles. For instance, EasyEnsemble [38] use bagging as the major ensemble method, but adopting AdaBoost in each bagging partition.

Even though ensemble learning methods make progress in dealing with imbalanced problems, there still exists some shortcomings. Ensemble learning methods, which combine the basic classifiers into a weight sum to represent the final output of the boosted classifiers, is nonintuitive. Some classifiers, such as geometry-based ensembles [39] and convex hull [40], are designed according to the geometric property. However, they are with low efficiency. In addition, the ensemble learning expects the basic classifiers to be with the properties of difference and diversity [13]. Furthermore, more basic classifiers cause higher computation complexity. For instance, higher number may get better result but with higher training time. What is more, there is no criterion to define the number of basic classifiers.

To overcome the problems mentioned above, a different ensemble strategy is designed in this paper. The proposed strategy is designed from the geometric perspective and is with high efficiency. In the course of training, the basic classifier partitions the minority and majority domains. Samples in the minority domain are decided as the minority samples, while the samples in majority domain are decided as the majority samples. Then, the majority domain and the samples inside it are removed. After that, the remained samples are used to train the next basic classifier until all majority samples have been removed. Finally, the remained area, which can be seen as the intersection of all minority domains partitioned by respective basic classifier, is the ensemble minority domain representing the decision region of minority class. To improve the generalization ability, the imbalanced data set with relatively higher imbalanced ratio (IR) is randomly divided into two subsets to make the majority sample sparse. In addition, the basic classifiers whose recognition ability is weak are pruned.

To cooperating with the training, the basic classifier has to recognize all minority samples and part of majority samples. By doing so, the recognized majority samples can be removed and the intersection of all minority domains contains all minority samples. Through introducing supporting hyperplane theorem, the basic classifier is designed in accordance with the geometric property of hyper-sphere. The property is that the tangent line of the hyper-sphere ensures that the hyper-sphere is on one side of the tangent line. Through fitting the majority samples as hyper-sphere and constructing one or more tangent lines, the basic classifier partitions the minority and majority domain. The minority domain contains all minority samples (it may contain some majority samples) and the majority domain just contains the majority samples, which means that the basic classifier recognizes all minority samples and part of majority samples.

As a result, a geometric structural ensemble (GSE) method is proposed in this paper with the abbreviation GSE for the sake of convenience. GSE can be distinguished from the existing imbalanced-oriented ensemble learning in terms of the special features listed below.

1) GSE gives an ensemble strategy based on spatial geometric partition rather than traditional variable-weight combination so as to achieve better interpretability. In addition, GSE continually reducing the training set in iterations, thus boosting the training speed and ensuring the diversity of basic classifiers.
2) GSE designs the geometric structural basic classifier, which is easy to implement. Therefore, the training speed is extremely fast. In addition, there is no need to preset the number of basic classifiers.
3) GSE utilizes two relaxation techniques to improve the generalization. One is randomly dividing the majority samples to make the distribution of majority samples sparse. The other is pruning useless basic classifiers to expand the region of the ensemble minority domain.

The rest of this paper is organized as follows. Section II introduces the overall process of the proposed GSE. Section III reports all the experimental results. Finally, the conclusion is given in Section IV.

II. Proposed Geometric Structural Ensemble Learning

This section presents the complete construction of the proposed GSE. Specifically, there are three main processes, including the ensemble strategy, the basic classifier and the relaxation techniques, to train GSE. Each of the processes is described in the following parts of this section.

A. Ensemble Strategy

In the beginning of the section, the imbalanced problem is defined as a training set that contains n samples-based vector form with d dimensions. For each sample x_i, it can be described as $x_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}] \in \mathbb{R}^d$, $i = 1, 2, \ldots, n$, d is the dimensions of the sample space. Suppose the number of minority samples is $n_{\min}$. Then, the minority samples are defined as $X_{\min} = \{x_{1_{\min}}, x_{2_{\min}}, \ldots, x_{n_{\min}}\} \in \mathbb{R}^d$, meanwhile majority samples are defined as $X_{\text{maj}} = \{x_{n_{\min}+1}, x_{n_{\min}+2}, \ldots, x_{n_{\min}+n_{\text{maj}}}\} \in \mathbb{R}^d$, where n_{maj} is the number of majority samples. In the imbalanced problem, n is equal to $n_{\min} + n_{\text{maj}}$ and the IR is defined as $(n_{\text{maj}}/n_{\min})$.

Under the framework of GSE, the basic classifier partitions two complementary decision regions representing the minority domain (*MinDomain*) and majority domain (*MajDomain*), respectively. Samples are decided as minority samples in the *MinDomain*, while decided as majority samples in the *MajDomain*. For every basic classifier, GSE requires that *MinDomain* contains all minority samples and does not concern whether it contains majority samples. Moreover, GSE requires that *MajDomain* contains at least one majority sample

TABLE I
LEARNING PROCESS: GSE

1. i = 0;
2. $EnsembleDomain \in \mathbb{R}^d$;
3. while $X_{maj} \neq \varnothing$
4. i = i + 1;
5. Train $Classifier^i$ on $\{X_{maj} \bigcup X_{min}\}$;
6. Record $MinDomain^i$ and X_{del_maj};
7. $EnsembleDomain = EnsembleDomain \bigcap MinDomain^i$;
8. $X_{maj} = X_{maj} \setminus X_{del\_maj}$;
9. end while

and none of minority samples. Then, the *MajDomain* and the majority samples inside *MajDomain* are removed. After that, the remained majority samples is used to train the next basic classifier. By doing so, every basic classifier naturally trains on different training data. Therefore, the basic classifier is naturally with the property of difference and diversity. When the majority samples set is empty, the remained area represents the ensemble minority domain (*EnsembleDomain*), which contains all minority samples and none of majority samples. The process of training basic classifiers can be summarized in Table I.

In practice, only the *MinDomain* is recorded in every iteration. The reason is that the target of GSE is to enclose the *EnsembleDomain*. The process of removing *MajDomains* can be represented by the intersection of all *MinDomains* partitioned by their corresponding basic classifiers. It is worth notice that the boundary between *MinDomain* and *MajDomain* is put under the *MajDomain*. In Table I, Classifier^i represents the ith basic classifier. The $MinDomain^i \in \mathbb{R}^d$ represents the ith *MinDomain* partitioned by Classifier^i and d is the dimensions of the sample space. $X_{\text{del\_maj}}$ contains the majority samples who are inside the $MajDomain^i$. These majority samples will be removed for the training of the next basic classifier.

According to Table I, *EnsembleDomain* is initialized as $\mathbb{R}^d$. First, a basic classifier $Classifier^1$ is trained on the original sample set $\{X_{\text{maj}} \bigcup X_{\text{min}}\}$ and it partitions the $MinDomain^1$ and $MajDomain^1$. Then, the samples inside the $MajDomain^1$ are regarded as $X_{\text{del\_maj}}$, which will be removed from the X_{maj}. Through continually intersecting the *MinDomains* in every "while" loop, *EnsembleDomain* encloses all minority samples and excludes all majority samples.

To explain it clearly, Fig. 1 shows the detailed training process of GSE. Fig. 1(b) is corresponding to the first basic classifier classifier^1. In the figure, the shadow area is the $MinDomain^1$ partitioned by classifier^1. It is found that $MinDomain^1$ contains all minority samples, which means that the minority samples are classified rightly by the basic classifier. In addition, the $MajDomain^1$ without shadow area contains part of majority samples and none of minority samples. Then, the majority samples inside the $MajDomain^1$ are removed for the training of the next basic classifier. It is clear that the samples inside the $MajDomain^1$ in Fig. 1(b) are missing in Fig. 1(c). The remained samples in Fig. 1(c) are used to train the second basic classifier classifier^2. Similarly, Fig. 1(e), (f), and (i)–(l) are corresponding to the following basic classifiers. Fig. 1(d) shows that the relationship of *MinDomains* corresponding to different basic classifier is intersection. Finally, through intersecting these *MinDomains* corresponding to all basic classifiers, the *EnsembleDomain* is shown in Fig. 1(m). In Fig. 1(m), the *EnsembleDomain* is the shadow area which contains all minority samples and none of majority samples.

B. Basic Classifier Generation

The requirement of the basic classifier under GSE framework is that the basic classifier has to recognize all minority samples and part of the majority samples. To meet the requirement, the supporting hyperplane theorem [41] is adapted as the theoretical basis in basic classifier designing. In detail, the corresponding theorem is described as follows.

Theorem 1: Suppose $S \in \mathbb{R}^n$ is a convex set, and x_0 is a point on the boundary of S. If f is a nonzero linear function, and $f(x_0) \geq f(x)$ is satisfied for all $x \in S$, then

$$\left\{x \middle| f(x) = f(x_0)\right\} \tag{1}$$

is called a supporting hyperplane to S at the point x_0. This is equivalent to say that the point x_0 and the set S are separated by the hyperplane. The geometric interpretation is that the supporting hyperplane H is tangent to the convex set S at x_0. And the half space $\{x|f(x) \leq f(x_0)\}$ includes S.

In GSE, the convex set is replaced by a hyper-sphere S, which only contains the majority samples. By introducing supporting hyperplane theorem, the majority samples in the hyper-sphere are on one side of the supporting hyperplane, which means these samples can be recognized. To ensure the location of the hyper-sphere, the mean point of minority samples is record as x_{mp}. After that, GSE searches the sample in the majority set (X_{maj}) and the sample who is farthest from x_{mp} is called the remote vertex (x_{rv}). Then, the remote vertex (x_{rv}) is regarded as the center of the hyper-sphere. To make the hyper-sphere only contain majority samples, GSE finds the minority sample closest to the remote vertex (x_{rv}). This sample is called the closest vertex (x_{cv}). As is shown in Fig. 1(a), the x_{mp} is marked as asterisk $*$. In addition, a hyper-sphere S of radius $r = d(x_{rv}, x_{cv})$ is represented and x_{rv} is the center. Here, $d(x_{rv}, x_{cv})$ is measured by Euclidean distance and X_{in_S} represents the majority samples covered by S. According to the supporting hyperplane theorem, the supporting hyperplane is tangent to the hyper-sphere S at x_{cv}. The norm vector $\vec{w}$ and threshold θ can be calculated as follows:

$$\vec{w} = x_{cv} - x_{rv} \tag{2}$$

$$\theta = -x_{cv} \times \vec{w}^T. \tag{3}$$

For every majority sample $x_i \in X_{in\_S}$ (in the hyper-sphere S) and the boundary sample x_{cv}

$$x_i \times \vec{w}^T + \theta \leq 0 \tag{4}$$

$$x_{cv} \times \vec{w}^T + \theta = 0. \tag{5}$$

In Fig. 1(a), the tangent hyperplane, described by dotted line, partitions the hyper-sphere on the negative side and the closest vertex x_{cv} is on the dotted line. However, the

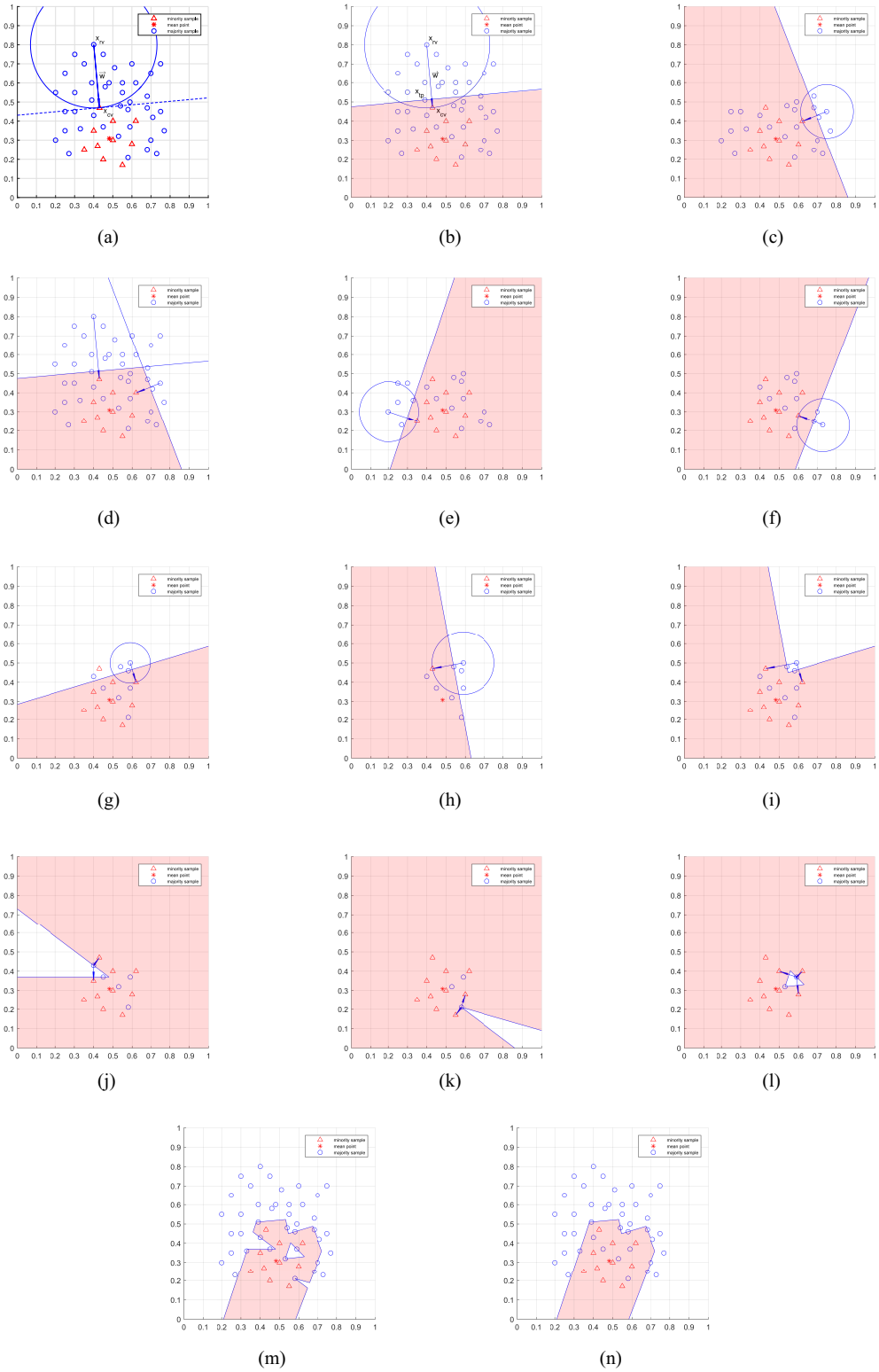

Fig. 1. Training process of GSE. (a) Classifier[1] with original hyperplane[1]. (b) Classifier[1] with modified hyperplane[1]. (c) Classifier[2]. (d) Combine classifier[1] and classifier[2] (intersection). (e) Classifier[3]. (f) Classifier[4]. (g) Classifier[5] (hyperplane[1]). (h) Classifier[5] (hyperplane[2]). (i) Classifier[5] (the union of hyperplane[1] and hyperplane[2]. (j) Classifier[6]. (k) Classifier[7]. (l) Classifier[8]. Combine all classifiers (m) without pruning and (n) with pruning.

classification hyperplane may be too oppressive for the minority class. To expand the region for minority class, the tangent hyperplane should be biased toward the samples $\in X_{in\_S}$. Therefore, majority samples $\in X_{in\_S}$ are projected onto the norm vector $\vec{w}$ and the sample with maximal projection value is the threshold point (x_{tp}), which can be written as

$$x_{tp} = \underset{x_i \in X_{in\_S}}{\operatorname{argmax}} (x_i - x_{rv}) \times \vec{w}^T. \tag{6}$$

TABLE II
ALGORITHM: CHG

Input: $X^*_{min} \subseteq X_{min}$, $X^*_{maj} \subseteq X_{maj}$, $x_{rv} \in X^*_{min}$, $x_{cv} \in X^*_{maj}$;
Output: the weight vector $\vec{w}$ and threshold θ of the hyperplane;
1. $\vec{w} = x_{cv} - x_{rv}$;
2. $X_{in\_S} = \{x_i \| d(x_i, x_{rv}) \le d(x_{cv}, x_{rv}), x_i \in X^*_{maj}\}$;
3. $x_{tp} = \text{argmax}_{x_i \in X_{in\_S}} (x_i - x_{rv}) \times \vec{w}^T$;
4. $\theta = -x_{tp} \times \vec{w}^T$;

Then, the new threshold θ can be rewritten as

$$\theta = -x_{tp} \times \vec{w}^T. \tag{7}$$

For $x_i \in X_{in\_S}$ and the boundary sample x_{cv}, it is obviously that the new hyperplane can ensures

$$x_i \times \vec{w}^T + \theta \le 0 \tag{8}$$

$$x_{cv} \times \vec{w}^T + \theta \ge 0. \tag{9}$$

The equality in the previous two inequalities implicates a rare case that x_{cv} is the same as x_{tp}, which means that the selected x_{rv} belongs to both minority class and majority class. In such a case, $\vec{w}$ is set as a nonzero vector to replace $x_{cv} - x_{rv} = 0$. Otherwise, classifier fails to give the boundary between *MinDomain* and *MajDomain*.

The actual operations can be seen in Fig. 1(b). Compared with the dotted line, the new hyperplane described by solid line is significantly more favorable to recognize minority samples. Until now, the classification hyperplane of the first basic classifier has been obtained. The shadow area represents the *MinDomain*. Outside the *MinDomain*, there just exits majority samples. The function about generating hyperplane is described in Table II.

Remember that GSE requires the *EnsembleDomain* contains all minority samples and none of majority samples. The *MinDomain* contains all minority samples, while *MajDomain* contains at least one majority sample and none of minority samples. According to Fig. 1(b), (c), (e), and (f), the corresponding classifiers[1,2,3,4] with one classification hyperplane reach the requirement. However, one hyperplane may fail to achieve the goal. As is shown in Fig. 1(g), the *MinDomian* partitioned by the first hyperplane hyperplane[1] of the fifth basic classifier classifier[5] fails to contain all minority samples. It is found that one minority sample is outside the *MinDomain*.

To make classifier[5] meet the requirement, another hyperplane should be added to discriminate the misclassified minority samples. Since the minority samples inside the *MinDomian* have been classified rightly, they are temporarily removed in current iteration. As is shown in Fig. 1(h), the minority samples inside the *MinDomian* partitioned by the first hyperplane hyperplane[1] has been removed. Then, the remained minority samples together with the majority samples are used to train another hyperplane. The detailed operation is that the center of the hyper-sphere (x_{rv}) finds another closest vertex (x_{cv}) in the remained minority samples. Next, the operation about getting $\vec{w}$ and θ of hyperplane[2] is the same as getting these parameters of hyperplane[1]. Then, a new hyper-sphere and corresponding hyperplane[2] are obtained. Fig. 1(h) shows that the remained minority samples are covered by *MinDomian* partitioned by hyperplane[2] and these samples will be temporarily removed. Until all minority samples have been removed, the union of the *MinDomians* partitioned by *hyperplanes* is the *MinDomain* of the current basic classifier. As is shown in Fig. 1(i), the *MinDomian* of *classifier*[5] is the union of the *MinDomians* partitioned by hyperplane[1] and hyperplane[2] and it reaches the design goal of the basic classifier. The formula about combining hyperplanes can be written as follows:

$$MinDomain^i = \bigcup_{j=1}^{H} \left\{ x \middle| x \times \text{classifier}^i.w_j^T + \text{classifier}^i.\theta_j > 0 \right\} \tag{10}$$

where $MinDomain^i$ is the minority domain of classifier^i, H is the number of hyperplanes, and classifier$^i.w_j$ and classifier$^i.\theta_j$ are the weight vector and threshold of the jth hyperplane of ith basic classifier. Finally, the temporarily removed minority samples have to be returned for the training of the next basic classifier. Similarly, classifiers[6,7,8] with more than one hyperplane can be obtained and they are shown in Fig. 1(j)–(l).

TABLE III
ALGORITHM: BCG

Input: x_{mp}, X_{min}, $X^*_{maj} \subseteq X_{maj}$;
Output: the *MinDomian* of the basic classifier and the removed majority samples set: X_{del_maj};
1. Initialization: $MinDomain = \varnothing$;
2. $x_{rv} = \text{argmax}_{x_i \in X^*_{maj}} d(x_i, x_{mp})$;
3. $X_{del\_maj} = X^*_{maj}$;
4. $X^*_{min} = X_{min}$;
5. While $X^*_{min} \neq \varnothing$
6. $x_{cv} = \text{argmax}_{x_i \in X^*_{min}} d(x_i, x_{rv})$;
7. $[\vec{w}, \theta] = CHG(X^*_{min}, X^*_{maj}, x_{rv}, x_{cv})$;
8. $MinDomain = MinDomain \cup \{x \| x \times \vec{w}^T + \theta > 0, x \in \mathbb{R}^d\}$;
9. $X_{del\_maj} = X_{del\_maj} \cap \{x \| x \times \vec{w}^T + \theta \le 0, x \in X^*_{maj}\}$;
10. $X_{del\_min} = \{x_i \| x_i \times \vec{w}^T + \theta \ge 0, x_i \in X^*_{min}\}$;
11. $X^*_{min} = X^*_{min} \setminus X_{del\_min}$;
12. End while

The complete pseudo code of basic classifier generation (BCG) is shown in Table III. In the table, $X^*_{\text{maj}} \subseteq X_{\text{maj}}$ is the remained majority samples for the training of the current basic classifier. Moreover, $X_{\text{del\_maj}}$, initialized as X^*_{maj}, is the majority samples which will be removed by the current basic classifier. Every hyperplane partitions its own *MinDomain*. Through continually merging the *MinDomain* partitioned by different hyperplane, the *MinDomain* of the basic classifier is obtained. Through continually intersecting the remained majority samples outside the *MinDomian* partitioned by different hyperplane, the $X_{\text{del\_maj}}$ is obtained. In the process of BCG, minority samples in X_{min} has to be temporarily removed. Therefore, X_{min} temporarily assigns to X^*_{min} and the operation about minority samples is done in X^*_{min}.

C. Relaxation Techniques for Ensemble Strategy

Although the *EnsembleDomain* contains all minority samples and none of majority samples, the generalization may be not enough. In practice, the *EnsembleDomain*, shown in Fig. 1(m), traps into the accuracy of training set, thus making the classification result worse on unknown test samples,

TABLE IV
ALGORITHM: GSE LEARNING

Input: the minority training set X_{min} and its size n_{min}, the majority training set X_{maj} and its size n_{maj};
Output: the merged minority class domain: $EnsembleDomain^U$;

1. Initialization: $EnsembleDomain^U = \varnothing$, P and R;
2. Divide the majority samples into P subsets;
3. $x_{mp} = \frac{1}{n_{min}} \sum_{i=1}^{n_{min}} x_i, x_i \in X_{min}$;
4. $IR = n_{maj}/(P \times n_{min})$ is the imbalanced ratio;
5. For i = 1 to P
6. $X^*_{maj} = X^i_{maj}$, X^i_{maj} is the *ith* majority sample subset;
7. j = 0;
8. Define $EnsembleDomain^i = \mathbb{R}^d$;
9. While $X^*_{maj} \neq \varnothing$
10. j = j + 1;
11. $[MinDomain^j, X_{del\_maj}] = BCG(x_{mp}, X_{min}, X^*_{maj})$;
12. If $|X_{del\_maj}| \geq R \times IR$;
13. $EnsembleDomain^i = EnsembleDomain^i \cap MinDomain^j$;
14. End if
15. $X^*_{maj} = X^*_{maj} \setminus X_{del\_maj}$;
16. End while
17. $EnsembleDomain^U = EnsembleDomain^U \cup EnsembleDomain^i$;
18. End for

especially for the minority samples. Therefore, two relaxation techniques are designed to overcome the problem.

One way is dividing the majority samples with high IR into two subsets. Once the majority samples are dividing into subsets, the distribution of majority subset is relatively sparser. Every majority subset together with the minority set constructs a new subtraining set. By doing so, the *EnsembleDomain* may be more advantage for the minority samples in the subtraining set. Here, $EnsembleDomain^i$ represents the *EnsembleDomain* learned in the *i*th subtraining set. Through merging all *EnsembleDomains* corresponding to different subtraining set, the final output of GSE $EnsembleDomain^U$ can be written as

$$EnsembleDomain^U = \bigcup_{i=1}^{P} EnsembleDomain^i \tag{11}$$

where P is the number of the subtraining sets.

The other way is pruning the unnecessary basic classifiers. Normally, basic classifiers, which remove only few majority sample, have little contribution to the final result. Moreover, these basic classifiers may cause over-fitting. Therefore, these basic classifiers are pruned. In Fig. 1, the IR of the synthetic data set is 4. Suppose the parameter P is equal to 1, which means that the original majority samples are without dividing. Here, a rational number R has to be set. Suppose R is set to 0.5, $4 \times 0.5 = 2$ is the threshold. Then, the basic classifier who removes equal or less than two majority samples is pruned. Therefore, *classifiers*[6,7,8] are pruned and the relaxed *EnsembleDomain* is shown in Fig. 1(n). Compared with Fig. 1(m), Fig. 1(n) shows a more comfortable *EnsembleDomain*, which has more advantages in recognizing minority samples. The pseudo code with relaxation techniques is shown in Table IV.

The decision function of GSE for the input sample x in testing process is obtained in accordance with the basic classifiers. For convenience, the set operation is transformed into arithmetic operation. First, the decision on hyperplane-level is given

$$H_{ij}(x) = \operatorname{sgn}\left(nH_{ij} + \sum_{k=1}^{nH_{ij}} \operatorname{sgn}\left(x \times \vec{w}^T_{ijk} + \theta_{ijk}\right)\right) \tag{12}$$

where $H_{ij}(x)$ is the decision value corresponding to the *j*th basic classifier in the *i*th subtraining set, nH_{ij} is the number of hyperplanes, and $\vec{w}_{ijk}, \theta_{ijk}$ is the *k*th weight vector and threshold of the *j*th basic classifier in the *i*th subtraining set. Obviously, $H_{ij}(x) \in \{0, 1\}$. Only when every hyperplane of the basic classifier judges x as majority sample, $H_{ij}(x)$ is equal to 0. Therefore, the union operation on hyperplane level has been completed. Further, the classifier-level is as follows:

$$C_i(x) = \prod_{j=1}^{nC_i} H_{ij}(x) \tag{13}$$

where $C_i(x) \in \{0, 1\}$ is the decision value corresponding to the *i*th subtraining set, and nC_i is the number of the basic classifiers. The value is equal to 0, only if at least one basic classifier determines the test sample x as majority sample. Finally, the complete decision function corresponding to $EnsembleDomain^U$ is written as

$$F(x) = \operatorname{sgn}\left(-0.5 + \sum_{i=1}^{P} C_i(x)\right) \begin{cases} > 0, & x \in \text{class}+ \\ < 0, & x \in \text{class}- \end{cases} \tag{14}$$

where P is the number of subtraining sets. In addition, class+ and class− are corresponding to the minority and majority class, respectively. Only if x is determined as minority sample in any of subtraining sets, it is a minority sample.

D. Convergence and Computational Complexity Analyses

According to Table IV, the convergence of the proposed GSE depends on two while loops. The terminal condition in the first while loop is $X^*_{\text{maj}} = \varnothing$. According to supporting hyperplane theorem, the samples partitioned by the basic classifier at least contains the remote vector x_{rv}. Therefore, the X^*_{maj} reduces in every loop until it becomes empty. The terminal condition of the second loop is in the function BCG. Similar to the first loop, the procedure ensures that the closest sample x_{cv} is removed, thus keeping the X^*_{min} down until X^*_{min} becomes empty. Then, the convergence of the proposed GSE is satisfied.

Next, we clarify that GSE is an $O(nd\log(n_{\text{min}})\log(n_{\text{maj}}))$ approximation algorithm, where n is equal to $n_{\text{min}}+n_{\text{maj}}$. In the algorithm, it contains the function BCG. In addition, another function classification hyperplane generation (CHG) is embed in the function BCG. We first analyze the inner function CHG shown in Table II. In detail, the time complexity is $O(d + n^*_{\text{maj}}d + n_{in\_S}d + d)$, where n^*_{maj} is the number of X^*_{maj} and n_{in_S} is the number of X_{in_S}. Since X^*_{maj} contains X_{in_S} and n_{in_S} is generally less than n^*_{maj}, the training time of the function is $O(n^*_{\text{maj}}d)$.

Then, we focus on the external function BCG shown in Table III. The running time of the second function concentrates on the while loop. The running times of the loop depend on the distribution of the data. Generally, more overlapping areas

require more running times. In fact, it is corresponding to the number of hyperplanes in the basic classifier. Inside the loop, it takes about $n^*_{\min}d+n^*_{\mathrm{maj}}d+d+n^*_{\mathrm{maj}}d+n^*_{\min}d+n^*_{\min}$ to run. Accordingly, the time complexity is $O(n^*_{\min}d+n^*_{\mathrm{maj}}d)$. Similar to the whole process in the quicksort [42]. the while loop can be converted into recursive mode. Therefore, the recursion for the running time is

$$T(n_{\min}) = T(n_{\min}-m)+m < T(n_{\min}-m)+O(n_{\min}) \quad (15)$$

where m is the number of temporarily removed minority samples and $1 < m < n_{\min}$. In best case, m is equal to $n_{\min}$ in the first recurrence. Then the running time of the function is $O(n_{\min}d+n^*_{\mathrm{maj}}d)$. In worst case, m is equal to 1 in every recurrence. The operation in ith loop needs $n_{\min}-i$ times to work and $O(\sum_i^{n_{\min}}(n_{\min}-i)) = O(n^2_{\min})$. Therefore, the running time of the function is $O(n^2_{\min}d+n_{\min}n^*_{\mathrm{maj}}d)$. For expected running time, similar with the random-partition in quicksort puts any constant fraction of the elements on one side of the partition, the depth of recursion tree is $O(\log(n_{\min}))$ and the total time on minority samples remains $O(n_{\min}\log(n_{\min}))$. Therefore, we can conclude the expected running time of the function is $O(n_{\min}d\log(n_{\min})+n^*_{\mathrm{maj}}d\log(n_{\min}))$.

Finally, we analyze the time complexity of the algorithm GSE shown in Table IV. The running time mainly focus on the "for" loop which further includes a while loop. Since the run times P of for loop is less than or equal to 2 and n_{maj} is reducing with P increasing, the for loop is ignored. Inside the for loop, the running time of the while loop mainly concentrates on the function BCG. Similar with the recursion in function CHG, the recursion for the running time is

$$T\big(n_{\mathrm{maj}}\big) = T\big(n_{\mathrm{maj}}-m\big)+m < T\big(n_{\mathrm{maj}}-m\big)+O\big(n_{\mathrm{maj}}\big) \quad (16)$$

where m is the number of removed majority samples and $1 < m < n_{\mathrm{maj}}$.

Similarly, the expected depth of recursion is $\log(n_{\mathrm{maj}})$ and the total time on majority samples is $O(n_{\mathrm{maj}}\log(n_{\mathrm{maj}}))$. Therefore, the expected running time of the proposed GSE is $O(n_{\min}d\log(n_{\min})\log(n_{\mathrm{maj}}) + n_{\mathrm{maj}}d\log(n_{\min})\log(n_{\mathrm{maj}}))$, which can be rewritten as $O(nd\log(n_{\min})\log(n_{\mathrm{maj}}))$.

E. Analysis of the Characteristics

To reflect the characteristic of the proposed GSE, another two representative ensemble methods, boosting and bagging, are briefly introduced. In boosting family, the AdaBoost is the most famous algorithm. It first assigns equal weights to all training samples. Suppose the weights in ith learning round is D_i. The algorithm generates a basic classifier h_i in accordance with the training data and D_i. Then, the weights of training samples who are incorrectly classified by h_i will be increased. By doings so, the importance α_i of h_i and the importance of the updated weights D_{i+1} is obtained. From the training data and D_{i+1}, AdaBoost trains another basic classifier h_{i+1}. Suppose the number of basic classifiers is T. For an input sample x, the output is equal to $\operatorname{sign}\sum_{i=1}^{T}\alpha_i h_i(x)$.

The bagging-based algorithms train basic classifier on different bootstrap samples sets obtained by randomly subsampling the train data with replacement. The size of the bootstrap samples set is predetermined and some samples may appear but some may not appear in the bootstrap samples set. When all basic classifiers are obtained, bagging combines all basic classifiers according to majority voting rule.

The proposed GSE first trains the basic classifier which partitions the minority domain (*MinDomain*) containing all minority samples and majority domain (*MajDomain*) containing at least one majority sample. Then, the majority samples in the *MajDomain* are removed from the training data. The remained training data is used to train the next basic classifier. The process continues until all majority samples are removed. Finally, through intersecting the *MinDomains* of all basic classifier, GSE gives the decision region of minority class. The union of *MajDomains* is the decision region of majority class.

Obviously, the GSE designs the ensemble strategy from the view of space intersection and union, which is quite different from the weight combination of boosting and majority voting of bagging. Moreover, GSE requires the basic classifier recognizes all minority samples and part of majority samples, while boosting and bagging require the basic classifier to be with better classification ability. In summary, the strategy combined with the basic classifier continuously recognizes the misclassified majority samples from the previous basic classifiers to ensure all samples classified rightly at last.

III. Experiments Setting

In this section, the experiments are designed to investigate effectiveness and efficiency of the proposed GSE. In detail, this section is divided into four parts. In the first part, the experiment settings including the used data sets, comparison groups, and basic settings are given. The classification results of all used algorithms are shown in the second part. The third part presents the training time of all used algorithms to reflect the efficiency of GSE. Finally, the influence of basic classifiers and parameter R is discussed.

A. Experimental Setup

1) Used Data Sets and Comparison Algorithms: To validate the performance of GSE on imbalanced problems, 50 imbalanced data sets from the knowledge extraction based on evolutionary learning (KEEL)[1] are used in the experiment. The information of all used data sets is shown in Table V. In the table, *Attribute* is the dimensions of the sample space. *Size* is the number of training samples and IR is the imbalanced ratio.

In order to validate the classification ability of the proposed GSE, another six classical algorithms are selected as comparison algorithms. These algorithms include optimized geometric ensemble (OGE) [39], EasyEnsemble [38], RUSBoost [32], ABRS-SVM (*L*), ABRS-SVM (*R*) [35], and RandomForest [43]. In ABRS-SVM, "*L*" and "*R*" means the "linear" kernel and "RBF" kernel. The reason why these algorithms are selected as comparison algorithms is that these algorithms are related to ensemble learning. In detail, OGE designs the ensemble learning method in accordance with the characteristic boundary points, which belongs to the

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

TABLE V
DATA SETS DESCRIPTION

Data sets	Attribute	Size	IR	Data sets	Attribute	Size	IR
ecoli0vs1	7	220	1.84	led7digit02456789vs1	7	443	11.21
glass1	9	214	1.85	glass06vs5	9	108	11.29
glass0	9	214	2.05	glass0146vs2	9	205	11.62
haberman	3	306	2.81	glass2	9	214	12.15
ecoli1	7	336	3.39	ecoli0147vs56	6	332	12.25
ecoli2	7	336	5.54	ecoli0146vs5	6	280	13.00
yeast3	8	1484	8.13	yeast1vs7	7	459	14.29
ecoli3	7	336	8.57	ecoli4	7	336	15.75
ecoli034vs5’;	7	200	9.00	glass4	9	214	16.10
ecoli0234vs5	7	202	9.06	page_blocks13vs4	10	472	16.14
yeast0359vs78	8	506	9.10	glass016vs5	9	184	20.00
ecoli046vs5	6	203	9.13	yeast1458vs7	8	693	22.08
ecoli0346vs5	7	205	9.25	yeast2vs8	8	482	23.06
ecoli0347vs56	7	257	9.25	shuttle_c2vs_c4	9	129	24.75
ecoli01vs235	7	244	9.26	yeast4	8	1484	28.68
yeast2vs4	8	514	9.28	yeast1289vs7	8	947	30.54
ecoli067vs35	7	222	9.41	poker9vs755tst	10	244	31.50
ecoli0267vs35	7	224	9.53	winequality_white9vs455tst	11	168	32.50
glass015vs2	9	172	9.54	yeast5	8	1484	32.91
yeast05679vs4	8	528	9.55	yeast6	8	1484	41.39
vowel0	13	988	9.97	ecoli0137vs26	7	281	43.80
ecoli067vs5	6	220	10.00	winequality_white3vs755tst	11	900	44.00
ecoli0147vs2356	7	336	10.65	poker89vs655tst	10	1485	58.40
glass016vs2	9	192	10.77	shuttle2vs555tst	9	3316	67.00
ecoli01vs5	6	240	11.00	abalone19	8	4174	127.42

optimal boundary. Therefore, OGE is with boundary geometric property. EasyEnsemble, embedding boosting into bagging, is a well performed algorithms in dealing with imbalanced problems. RUSBoost, combining bagging and boosting, is also an effective and efficient algorithm. For ABRS-SVM, though combining asymmetric bagging and random subspace, it makes considerable results in imbalanced problems. Finally, the RandomForest is a classical tree-based ensemble method.

In the experiments, GSE contains two parameters P and R. The parameter P, which controls the number of subtraining sets, is decided by the IR. Generally, it is set to 1. When the IR is bigger than 4, it is set to 2. Therefore, it is a fixed parameter. The parameter R, which controls the pruning technique, is selected from $\{0, 0.5, 1, 1.5, 2\}$. In OGE, the parameter c, which controls the regularization term, is selected from $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$. The setting of EasyEnsemble is consistent with the corresponding article [38], where the times of bagging are set to 4 and the times of boosting are set to 10. The boosting times of RUSBoost is set to 40. In EasyEnsemble and RUSBoost, the C 4.5 is selected as basic classifier. In ABRS-SVM, the slacking parameter c is selected from $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$. The dimensions of each attribute subset is set to 60% of original dimensions and the number of attribute subsets is set to 5. When the RBF kernel is used, the parameter σ is selected from $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$. Finally, the number of trees in RandomForest is setting to 100. For all used algorithms, the parameters are adjusted to achieve the highest performance in every data set.

2) Evaluation Criterion: In imbalanced problems, the confusion matrix shown in Table VI presents the relationship of samples recognized correctly or incorrectly. In this table, the positive class and negative class are corresponding to minority class and majority class, respectively. In this paper, GMean is selected as the evaluation criterion. It is calculated as follows:

$$\text{GMean} = \sqrt{\left(\frac{\text{TP}}{\text{TP}+\text{FN}}\right) \times \left(\frac{\text{TN}}{\text{TN}+\text{FP}}\right)}. \tag{17}$$

TABLE VI
CONFUSION MATRIX OF THE IMBALANCED PROBLEM

	Positive prediction	Negative prediction
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Negative (FP)	True Negative (TN)

It is the geometric mean of the correct rate for minority class and majority class. Beside the GMean results, the Friedman test [44] is used to detect the difference between all used algorithms. Moreover, the win/loss/tie count validates the superior performance of GSE. Furthermore, the broken line about true positive rate (TPR) and false positive rate (FPR) is listed.

In order to get the average validation performance on the imbalanced data sets, fivefold cross-validation strategy is adopted in the experiment. Each data set used in the experiments is split into five parts, where the one for testing and the others for training. All of the algorithms train on the same partitions of the training and test data. All the computations are performed on Intel i5 6600K with 3.50 GHz, 16G RAM DDR4, Microsoft Windows 10, and MATLAB R2015b environment.

B. Classification Results

The classification performances of GSE and the other comparison algorithms are listed in Table VII. It is found that ABRS-SVM (R) gets the first place in the average GMean and the proposed GSE get the second place. Although the average GMean of GSE is less than that of ABRS-SVM (R), the disparity is not obvious. The value of GSE is 0.25% less than that of ABRS-SVM (R). Compared with OGE, GSE relatively performs better. The average GMean of GSE is 0.60% higher than that of OGE. For the other comparison algorithms, GSE has obvious advantages in the value of GMean.

The average rank of Friedman test validates the performance of the proposed GSE. According to Table VII, it is found that the average rank of GSE is the lowest, which means that GSE

TABLE VII
TEST GMEAN (%), FRIEDMAN RANK, AND WIN/LOSS/TIE COUNT FOR GSE, OGE, EASYENSEMBLE, RUSBOOST, ABRS-SVM (*L*), ABRS-SVM (*R*), AND RANDOMFOREST ON KEEL DATA SETS (THE BEST RESULT ON EACH DATA SET IS WRITTEN IN BOLD)

Data set	GSE	OGE	EasyEnsemble	RUSBoost	ABRS-SVM (L)	ABRS-SVM (R)	RandomForest
	GMean ± std	GMean ± std	GMean ± std	GMean ± std	GMean ± std	GMean ± std	GMean ± std
ecoli0vs1	97.58 ± 1.93	97.33 ± 1.50	97.22 ± 1.99	**98.28 ± 1.70**	**98.28 ± 1.70**	97.95 ± 2.20	**98.28 ± 1.70**
glass1	73.99 ± 9.12	72.64 ± 7.75	**78.04 ± 6.12**	77.20 ± 7.16	63.11 ± 5.68	76.07 ± 5.22	75.21 ± 9.97
glass0	79.62 ± 4.74	77.98 ± 3.64	83.41 ± 5.17	82.84 ± 7.37	68.60 ± 4.82	80.00 ± 5.44	**84.41 ± 6.09**
haberman	58.75 ± 2.68	**64.62 ± 5.87**	62.64 ± 8.53	58.18 ± 4.32	57.70 ± 5.27	61.92 ± 6.54	44.46 ± 12.79
ecoli1	**89.99 ± 4.08**	88.36 ± 4.07	88.50 ± 3.79	89.11 ± 6.49	86.78 ± 3.34	86.97 ± 4.15	83.87 ± 4.68
ecoli2	87.72 ± 5.17	92.29 ± 4.29	90.15 ± 5.31	89.37 ± 5.88	86.99 ± 5.74	**93.14 ± 5.53**	85.84 ± 8.01
yeast3	89.63 ± 2.55	91.65 ± 2.08	**92.23 ± 1.48**	91.74 ± 3.34	91.16 ± 2.99	92.37 ± 2.44	82.54 ± 4.58
ecoli3	88.53 ± 2.56	**91.16 ± 2.30**	88.08 ± 2.95	87.21 ± 4.67	88.02 ± 4.58	89.60 ± 5.04	69.97 ± 11.71
ecoli034vs5';	91.67 ± 5.84	89.45 ± 13.12	**92.57 ± 6.00**	89.72 ± 13.41	86.44 ± 11.82	89.18 ± 6.51	85.21 ± 15.16
ecoli0234vs5	89.84 ± 7.67	90.33 ± 12.36	90.90 ± 12.60	**92.23 ± 12.91**	87.95 ± 13.94	86.98 ± 14.20	88.03 ± 11.89
yeast0359vs78	70.68 ± 2.59	**75.96 ± 4.75**	71.77 ± 7.87	68.36 ± 8.79	69.27 ± 3.15	60.72 ± 2.20	46.47 ± 4.48
ecoli046vs5	**91.83 ± 6.52**	91.74 ± 7.25	82.34 ± 10.85	87.21 ± 10.97	87.99 ± 13.45	88.09 ± 6.31	85.64 ± 10.30
ecoli0346vs5	90.36 ± 6.95	89.13 ± 7.22	89.64 ± 9.54	**93.62 ± 7.62**	91.19 ± 6.72	86.32 ± 8.80	85.60 ± 9.92
ecoli0347vs56	90.10 ± 11.33	**90.80 ± 9.66**	86.75 ± 8.30	88.86 ± 9.39	87.12 ± 10.01	87.12 ± 10.01	86.66 ± 5.16
ecoli01vs235	87.28 ± 4.54	85.46 ± 9.86	84.60 ± 9.99	79.79 ± 17.79	**87.45 ± 4.06**	85.81 ± 5.27	77.73 ± 20.80
yeast2vs4	92.83 ± 4.03	91.51 ± 2.61	**94.29 ± 3.94**	89.64 ± 3.36	89.27 ± 5.49	92.42 ± 4.11	85.13 ± 5.04
ecoli067vs35	84.57 ± 7.72	**84.60 ± 22.98**	79.12 ± 20.91	73.00 ± 40.95	80.70 ± 21.13	84.21 ± 14.08	74.70 ± 42.12
ecoli0267vs35	**87.81 ± 6.53**	83.73 ± 13.03	81.11 ± 12.61	81.91 ± 12.86	84.35 ± 8.84	83.97 ± 5.74	80.53 ± 12.69
glass015vs2	73.47 ± 14.69	75.04 ± 10.59	67.66 ± 11.18	55.41 ± 35.16	**79.91 ± 10.74**	73.75 ± 10.76	16.33 ± 36.51
yeast05679vs4	**81.67 ± 5.84**	80.52 ± 6.29	79.43 ± 5.89	75.20 ± 11.46	80.21 ± 3.90	81.07 ± 4.06	56.50 ± 17.95
vowel0	97.17 ± 0.88	**98.99 ± 0.32**	97.69 ± 1.28	95.96 ± 2.63	83.57 ± 3.23	90.95 ± 1.48	93.94 ± 7.49
ecoli067vs5	87.83 ± 5.50	**87.98 ± 7.63**	83.39 ± 5.19	83.12 ± 9.06	84.81 ± 8.45	84.81 ± 8.45	84.42 ± 20.47
ecoli0147vs2356	**88.85 ± 5.06**	87.98 ± 7.67	83.90 ± 7.37	86.32 ± 7.39	82.24 ± 5.77	85.14 ± 5.75	82.40 ± 8.05
glass016vs2	77.64 ± 13.63	65.48 ± 12.82	69.76 ± 11.42	67.39 ± 12.76	74.80 ± 7.30	**78.77 ± 5.49**	21.55 ± 29.63
ecoli01vs5	**91.79 ± 7.12**	90.27 ± 7.48	81.99 ± 13.84	84.79 ± 10.90	89.02 ± 8.10	91.66 ± 6.14	88.16 ± 11.98
led7digit02456789vs1	87.29 ± 6.47	85.89 ± 5.44	87.65 ± 9.28	**89.45 ± 8.09**	85.10 ± 5.80	87.56 ± 5.87	88.85 ± 9.77
glass06vs5	93.10 ± 3.77	92.77 ± 13.39	90.88 ± 5.00	91.06 ± 11.60	88.70 ± 2.54	88.14 ± 2.65	**94.14 ± 13.10**
glass0146vs2	**81.25 ± 6.03**	75.70 ± 10.24	69.81 ± 4.74	81.09 ± 7.39	74.87 ± 6.71	79.56 ± 8.69	10.00 ± 22.36
glass2	78.86 ± 8.18	73.80 ± 9.64	75.21 ± 7.61	76.50 ± 20.65	81.82 ± 9.64	**82.53 ± 9.54**	0.00 ± 0.00
ecoli0147vs56	90.34 ± 3.43	**90.84 ± 4.22**	88.84 ± 5.05	89.43 ± 4.71	88.81 ± 3.94	88.57 ± 4.28	83.42 ± 14.29
ecoli0146vs5	**91.57 ± 12.98**	88.99 ± 11.67	91.20 ± 5.96	86.85 ± 11.76	89.29 ± 12.39	90.69 ± 6.57	83.56 ± 22.42
yeast1vs7	73.61 ± 7.58	75.28 ± 1.54	74.94 ± 9.41	68.60 ± 5.85	**78.34 ± 6.80**	78.65 ± 8.52	34.50 ± 31.50
ecoli4	94.71 ± 6.98	**97.75 ± 0.90**	89.19 ± 7.64	89.42 ± 7.43	91.22 ± 2.04	91.92 ± 1.09	84.48 ± 20.34
glass4	89.06 ± 12.28	89.68 ± 11.62	82.13 ± 9.40	87.34 ± 12.33	**93.29 ± 2.49**	90.31 ± 2.44	63.58 ± 38.58
page_blocks13vs4	96.09 ± 0.77	93.81 ± 2.49	96.38 ± 3.63	**99.44 ± 0.57**	90.50 ± 2.97	92.97 ± 2.62	96.04 ± 5.44
glass016vs5	93.48 ± 2.30	87.19 ± 13.27	92.17 ± 4.91	**94.68 ± 3.07**	90.60 ± 4.95	90.88 ± 5.84	53.85 ± 50.53
yeast1458vs7	**67.52 ± 8.98**	64.08 ± 5.62	59.28 ± 8.16	60.33 ± 11.18	63.45 ± 4.09	64.31 ± 5.94	0.00 ± 0.00
yeast2vs8	**80.08 ± 9.58**	76.32 ± 11.49	79.47 ± 5.40	77.24 ± 7.64	77.03 ± 8.57	74.35 ± 8.81	69.65 ± 12.86
shuttle_c2vs_c4	96.10 ± 5.57	99.60 ± 0.90	80.00 ± 44.72	80.00 ± 44.72	**100.00 ± 0.00**	**100.00 ± 0.00**	94.14 ± 13.10
yeast4	83.85 ± 4.28	83.61 ± 3.97	83.06 ± 7.36	**84.33 ± 8.25**	82.32 ± 3.89	81.06 ± 2.63	40.12 ± 13.95
yeast1289vs7	70.33 ± 4.91	**74.20 ± 4.92**	72.94 ± 8.20	62.79 ± 14.29	72.76 ± 3.12	72.73 ± 2.21	19.67 ± 27.60
poker9vs755tst	65.25 ± 38.55	63.40 ± 37.58	**73.96 ± 16.64**	18.22 ± 40.74	64.33 ± 9.86	66.45 ± 9.64	14.14 ± 31.62
winequality_white9vs455tst	52.46 ± 48.04	67.55 ± 38.05	0.00 ± 0.00	0.00 ± 0.00	**70.06 ± 39.39**	68.26 ± 38.56	20.00 ± 44.72
yeast5	97.11 ± 0.63	96.68 ± 0.46	**97.22 ± 0.54**	95.99 ± 3.18	96.54 ± 0.66	96.07 ± 0.84	83.43 ± 6.38
yeast6	**88.81 ± 6.92**	88.50 ± 5.64	87.18 ± 7.12	83.22 ± 7.73	87.59 ± 5.87	88.32 ± 5.55	60.83 ± 18.77
ecoli0137vs26	76.63 ± 42.89	68.64 ± 39.55	85.33 ± 4.72	71.21 ± 41.30	**89.31 ± 4.13**	88.03 ± 5.14	74.01 ± 43.36
winequality_white3vs755tst	**73.56 ± 3.01**	59.36 ± 34.07	71.36 ± 12.18	71.20 ± 7.67	69.42 ± 7.94	71.35 ± 9.07	30.00 ± 27.39
poker89vs655tst	51.40 ± 5.01	50.02 ± 13.63	73.62 ± 14.11	84.80 ± 15.32	44.20 ± 10.80	**85.08 ± 7.82**	0.00 ± 0.00
shuttle2vs555tst	99.72 ± 0.17	99.69 ± 0.13	99.15 ± 0.28	99.94 ± 0.14	99.69 ± 0.12	**100.00 ± 0.00**	**100.00 ± 0.00**
abalone19	75.49 ± 10.84	**80.53 ± 5.97**	71.73 ± 7.64	61.88 ± 10.85	75.95 ± 3.94	74.70 ± 4.67	0.00 ± 0.00
average GMean	83.78 ± 8.18	83.18 ± 9.47	81.40 ± 8.08	79.43 ± 11.50	82.24 ± 6.86	**84.03 ± 6.30**	63.24 ± 15.94
average rank	**2.77**	3.46	3.92	4.22	4.20	3.42	5.99
Win/Loss/Tie	-/-/-	31/19/0	32/18/0	37/13/0	37/13/0	30/20/0	44/6/0

has advantages in most of the used data sets. According to Friedman test, the value of τ_F is calculated and it is equal to 13.48. Compared with the critical value of $F(7-1, (7-1)(50-1)) = 2.13$ under the confidence level of 0.95, the $\tau_F = 13.48$ is larger. Therefore, the performance of these algorithms is significantly different. Moreover, the Nemenyi *post-hoc* test is used to assess the difference between two algorithms. The value q_α is 2.949 when there are seven algorithms with $\alpha = 0.05$. Therefore, the critical difference (CD) in Nemenyi *post-hoc* test is equal to 1.27.

It means that two algorithms are obviously different when the difference of average rank is over 1.27. According to Table VII, GSE is quite different from algorithms including RUSBoost, ABRS-SVM (*L*), and RandomForest, since the corresponding values of these algorithms are over $2.77 + 1.27 = 4.04$. The average value of EasyEnsemble is close to 4.04, which demonstrates that GSE also obviously outperforms EasyEnsemble. For the comparison algorithms including OGE and ABRS-SVM (*R*), the average ranks of these two algorithms achieve 3.46 and 3.42. The difference between GSE and the two comparison algorithms is still higher than half of the value of CD. Therefore, GSE performs better than OGE and ABRS-SVM (*R*). In addition, the win/loss/tie count also validates that GSE has better performance than the other comparison algorithms. According to Table VII, it is obvious that GSE gets advantages in more data sets when it compares with the other comparison algorithms.

To compare the recognition rate of minority and majority samples, the broken line about TPR and FPR is listed in Fig. 2. Here, the 50 data sets are divided into fivefold according to the IR from low to high and each fold contains ten data sets. Then, the average value of TPR and FPR in every fold is pointed in

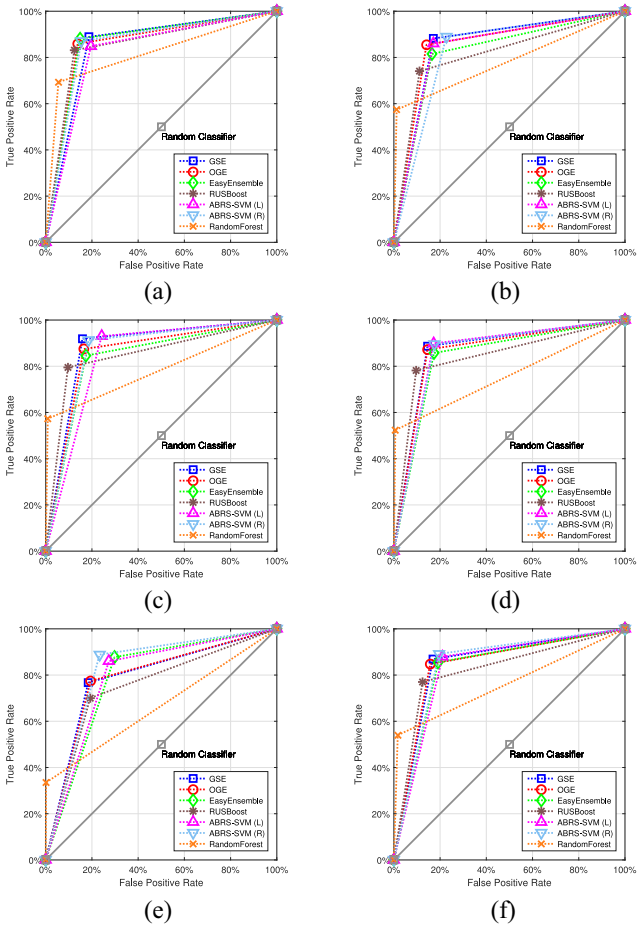

Fig. 2. Broken line about TPR and FPR for all used comparison algorithms. Data sets from (a) 1 to 10 (average), (b) 11 to 20 (average), (c) 21 to 30 (average), (d) 31 to 40 (average), (e) 41 to 50 (average), and (f) 1 to 50 (average).

the figure. The classifier whose point about TPR and FPR is closer to the top left corner is with better classification ability, since the area under the broken line is relatively larger.

In Fig. 2(a), these data sets are with relatively lower IR. EasyEnsemble is with relatively obvious advantages. The performance of GSE is competitive with RUSBoost and slightly worse than OGE and ABRSSVM (R). In Fig. 2(b), these data sets are with relatively higher IR than that in Fig. 2(a). It is found that GSE and OGE are competitive and these two algorithms perform better than the other algorithms. In detail, GES has stronger recognition for minority samples, while OGE has stronger recognition for majority samples. In Fig. 2(c), these data sets are with medium IR. Obviously, GSE outperforms the other comparison algorithms. In Fig. 2(d), these data sets are with relatively higher IR than that in Fig. 2(c). It is found that GES also outperforms the other comparison algorithms. Finally, in Fig. 2(d), the ABRS-SVM (R) outperforms the other algorithms. GSE is competitive with OGE, ABRS-SVM (L), and EasyEnsemble.

In detail, GSE has obvious advantages in performance when IR of the data set is medium (about 20 IR). When the data set is with lower IR, GSE is with higher value in TPR and FPR, which means that GSE tends to recognize more minority samples but misclassify more majority samples. When the data set is with higher IR, the trend of GSE is opposite. The pruning technique accounts for the phenomenon. With the IR of data set increasing, the pruning threshold is increasing. Thus, the candidated pruned majority domain is larger and with more majority samples. Therefore, GSE can not casually remove these domains. The more preserved majority domains may press the decision region of minority class, thus reducing the TPR and FPR, which means that GSE recognizes more majority samples but less minority samples with IR of data set increasing. Generally, as is shown in Fig. 2(f), the classification ability of GSE and ABRS-SVM (R) are competitive. Compared with OGE, GSE has advantages in most data sets. Moreover, GSE outperforms the other comparison algorithms.

TABLE VIII
PEARSON CORRELATION COEFFICIENT BETWEEN THE NUMBER OF BASIC CLASSIFIERS AND THE OTHER PROPERTY OF DATA

Correlation Coefficient	Attribute	Size	IR	Size/Attribute
Number of classifiers	0.05	0.58	0.41	0. 61

C. Training Time Comparison

The proposed GSE is an algorithm with high efficiency and the time complexity is $O(nd\log(n_{\min})\log(n_{\text{maj}}))$. In this section, the training time of all used algorithms is list in Fig. 3. Since OGE, ABRS-SVM (L) and ABRS-SVM (R) may consume more time in training and the GSE consumes few time, the training time is limited to 1 s to make the difference more obvious. The training time is corresponding to the best results shown in Table VII.

In Fig. 3, it is obvious that the required training time of GSE achieves the lowest in all data sets. There is no doubt that the proposed GSE is with very high training speed. Although the classification ability between GSE and ABRS-SVM (R) is comparable, the training time of GSE is far less than that of ABRS-SVM (R) whose time complexity is $O(\text{IR}(n_{\min}d)^3)$. With the IR increasing, the training time of ABRS-SVM (R) is increasing. Since OGE and ABRS-SVM (L) also use asymmetric bagging as ABRS-SVM (R) do, the time consuming of themselves is similar to that of ABRS-SVM (R). For EasyEnsemble, RUSBoost, and RandomForest, these algorithms use tree-based basic classifier. Therefore, the time complexity of basic classifier is $O(n^2d\log(n))$. In detail, the number of basic classifiers is equal to 40 in EasyEnsemble and RUSBoost. Therefore, the time complexities of these two algorithms are $O(40\times n_{\min}^2 d\log(n_{\min}))$. From the viewpoint of orders of magnitude, $n_{\min}^2$ is bigger than $n_{\min}(\text{IR}+1)\log(n_{\min}\text{IR})$, which is required in GSE. Therefore, the training speed of GSE is faster than that of EasyEnsemble and RUSBoost. Similarly, RandomForest, using the decision tree as basic classifier, consumes more time than GSE in training process.

D. Influence of Basic Classifiers and Parameter

Here, the relationship between the number of basic classifiers and the other property of data is listed in Table VIII. According to the table, the number of basic classifiers is

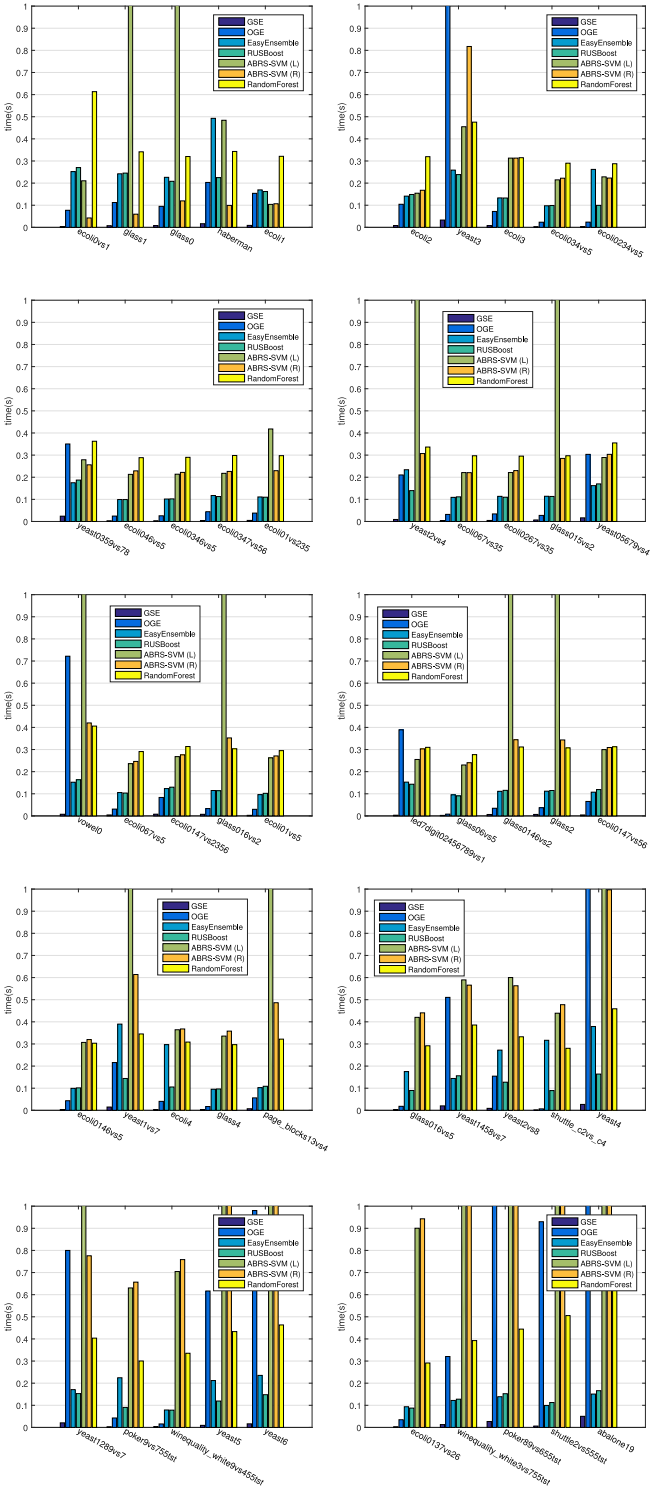

Fig. 3. Training time of GSE and the other comparison algorithms on all used data sets.

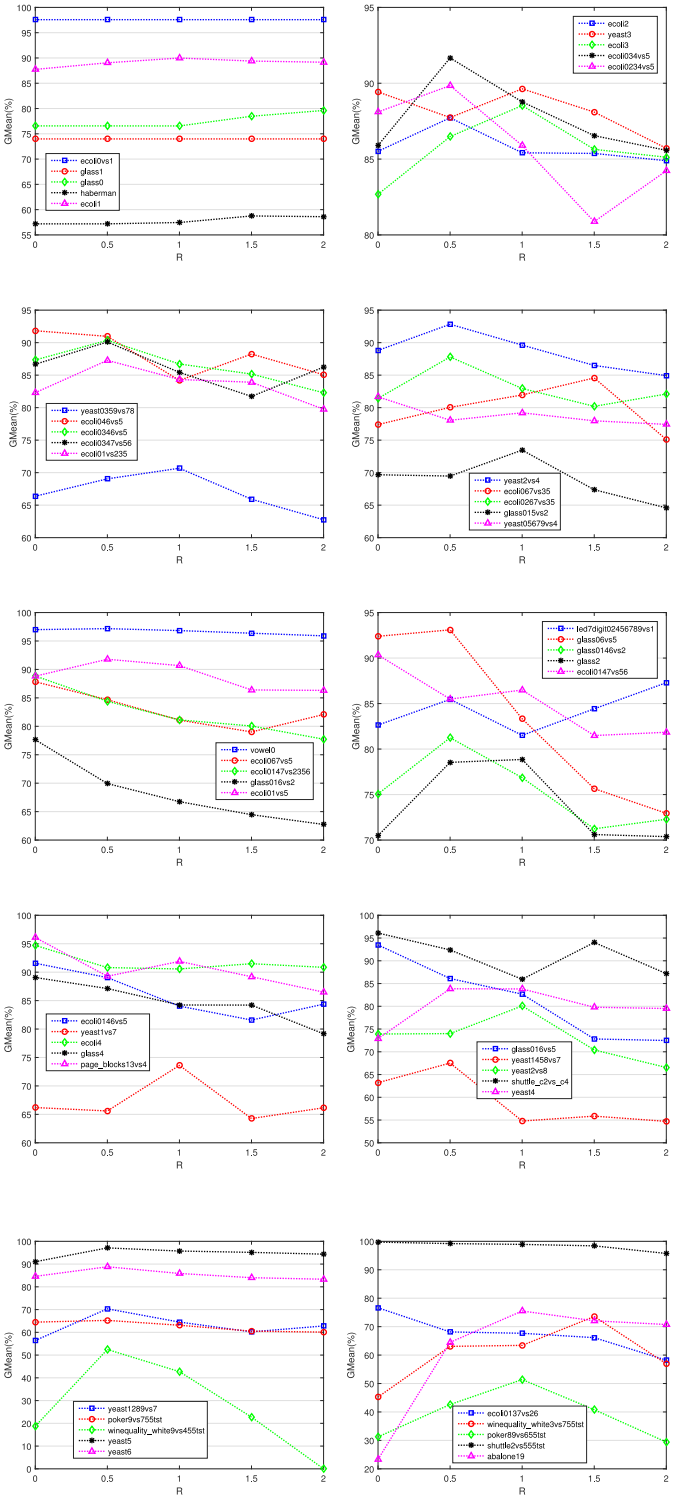

Fig. 4. Test classification results of GSE with varying R on all used data sets.

irrelevant to the *Attribute* of the data. Moreover, it has medium correlation with the IR. It is worth noticing that the number of basic classifiers has relatively stronger correlation with the *Size* and the *Size*/*Attribute* of the data. The reason is that the basic classifier is designed in accordance with the location of samples. Generally, the more samples with higher IR and less attributes exist more overlapping area. Therefore, this situation needs more basic classifiers.

To avoid overfitting on the training set, the basic classifier who removes few majority samples are pruned. In the proposed GSE, there is a varying parameter R, which decides whether the basic classifier is pruned. Through multiplying R and IR, the threshold value of samples eliminated by basic classifier is obtained. If the number of majority samples eliminated by basic classifier is less than IR $\times R$, the basic classifier is pruned.

As is shown in Fig. 4, the classification results of most data sets are improved through pruning some weak classifiers. 11 data sets including ecoli046vs5, yeast05679vs4, ecoli067vs5, ecoli0147vs2356, glass016vs2, ecoli0147vs56, ecoli0146vs5, ecoli4, glass4, glass016vs5, and ecoli0137vs26, do not get benefits from the pruning technique. Generally, these data sets, especially for the ecoli-type data sets, are with relatively less overlapping area and few noise samples. Therefore, excessive pruning may partition too many majority domains to the minority decision region, thus reducing the recognition rate of majority samples. For the other data sets, these data sets are with overlapping areas. Generally, minority samples in overlapping area need more attentions. Therefore, pruning some weak classifiers brings benefits for the recognition for minority samples. In practice, the irregular distributions, especially for the overlapping situation, need the pruning technique, while the data sets with less overlapping area do not.

The major defect is that the design of the relaxation techniques is in accordance with the experience, thus being lake of theoretical guidance. Some pruned basic classifiers may play important role in classification, especially in the complex situation. In the future, the error functions, which represent the properties of the pruned area, should be designed to make the relaxation technique more robust.

IV. Conclusion

This paper analyzes the deficiency of the existing ensemble learning methods in dealing with imbalanced problems and proposes a GSE learning framework. In detail, the advantages mainly concentrate in three major aspects including a strategy of ensemble learning, a designing of basic classifier, and two relaxation techniques. In ensemble strategy, strategy of GSE is based on space partition, thus having better interpretability. In basic classifier designing, the property of hyper-sphere makes the basic classifier geometric intuitive. Moreover, continuously removing sampling for training basic classifier greatly reduces the training time. In relaxation techniques, one is dividing the majority samples into subtraining sets to making the majority samples sparse. The other is pruning relatively useless basic classifiers to expand the decision region of minority class. Theoretically, the time complexity of GSE is $O(nd\log(n_{\min})\log(n_{\mathrm{maj}}))$. The experiments validate both effectiveness and efficiency. Thus, we can conclude that GSE is a not only effective but also fast algorithm in dealing with imbalanced problems.

References

[1] M. Zieba, "Service-oriented medical system for supporting decisions with missing and imbalanced data," *IEEE J. Biomed. Health Inf.*, vol. 18, no. 5, pp. 1533–1540, Sep. 2014.

[2] X. Yuan, L. Xie, and M. Abouelenien, "A regularized ensemble framework of deep learning for cancer detection from multi-class, imbalanced training data," *Pattern Recognit.*, vol. 77, pp. 160–172, May 2018.

[3] Z. Yang, W. H. Tang, A. Shintemirov, and Q. H. Wu, "Association rule mining-based dissolved gas analysis for fault diagnosis of power transformers," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 39, no. 6, pp. 597–610, Nov. 2009.

[4] H. He and E. Garcia, "Learning form imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[5] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[6] G. E. A. P. A. Batista, R. Prati, and M. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 20–29, 2004.

[7] O. Loyola-González, M. García-Borroto, M. Medina-Pérez, J. Martínez-Trinidad, and J. Carrasco-Ochoa, "An empirical study of oversampling and undersampling methods for LCMine an emerging pattern based classifier," in *Proc. Mexican Conf. Pattern Recognit.*, Lect. Notes Comput. Sci., vol. 7914, 2013, pp. 264–273.

[8] S. García, J. Derrac, J. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 417–435, Mar. 2012.

[9] N. V. Chawla, D. A. Cieslak, L. O. Holl, and A. Josh, "Automatically countering imbalance and its empirical relationship to cost," *Data Min. Knowl. Disc.*, vol. 17, no. 2, pp. 225–252, 2008.

[10] S. Zhang, L. Liu, X. Zhu, and C. Zhang, "A strategy for attributes selection in cost-sensitive decision trees induction," in *Proc. IEEE Int. Conf. Comput. Inf. Technol. Workshops*, 2008, pp. 8–13.

[11] C. X. Lin, V. S. Sheng, and Q. Yang, "Test strategies for cost-sensitive decision trees," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 8, pp. 1055–1067, Aug. 2006.

[12] Z. Sun *et al.*, "A novel ensemble method for classifying imbalanced data," *Pattern Recognit.*, vol. 48, no. 5, pp. 1623–1637, 2015.

[13] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 4, pp. 463–484, Jul. 2014.

[14] N. V. Chawla, K. W. Bowyer, L. O. Holl, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.

[15] S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 405–425, Feb. 2014.

[16] M. Pérez-Ortiz, P. A. Gutiérrez, P. Tino, and C. Hervás-Martánez, "Oversampling the minority class in the feature space," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 9, pp. 1947–1961, Sep. 2016.

[17] B. Sun, H. Chen, J. Wang, and H. Xie, "Evolutionary under-sampling based bagging ensemble method for imbalanced data classification," *Front. Comput. Sci.*, vol. 12, no. 2, pp. 331–350, Apr. 2018.

[18] W.-C. Lin, C.-F. Tsai, Y.-H. Hu, and J.-S. Jhang, "Clustering-based undersampling in class-imbalanced data," *Inf. Sci.*, vols. 409–410, pp. 17–26, Oct. 2017.

[19] F. Yuan, J.-H. Liang, Y.-W. Fu, H.-C. Xu, and K. Ma, "A hybrid sampling strategy with optimized probabilistic roadmap method," in *Proc. Int. Conf. Fuzzy Syst. Knowl. Disc.*, 2016, pp. 2298–2302.

[20] Y. Yang and J. Jiang, "Hybrid sampling-based clustering ensemble with global and local constitutions," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 5, pp. 952–965, May 2016.

[21] X.-Y. Liu and Z.-H. Zhou, "The influence of class imbalance on cost-sensitive learning: An empirical study," in *Proc. Int. Conf. Data Min.*, 2006, pp. 970–974.

[22] C. L. Castro and A. P. Braga, "Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 6, pp. 888–899, Jun. 2013.

[23] W. W. Y. Ng, J. Hu, D. S. Yeung, S. Yin, and F. Roli, "Diversified sensitivity-based undersampling for imbalance classification problems," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2402–2412, Nov. 2017.

[24] M. Liu *et al.*, "Cost-sensitive feature selection by optimizing F-measures," *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1323–1335, Mar. 2018.

[25] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Hoboken, NJ, USA: Wiley, 2001.

[26] B. Wang and J. Pineau, "Online bagging and boosting for imbalanced data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp. 3353–3366, Dec. 2016.

[27] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "AdaCost: Misclassification cost-sensitive boosting," in *Proc. 16th Int. Conf. Mach. Learn.*, 1999, pp. 97–105.

[28] M. Joshi, V. Kumar, and R. C. Agarwal, "Evaluating boosting algorithms to classify rare classes: Comparison and improvements," in *Proc. IEEE Int. Conf. Data Min.*, 2001, pp. 257–264.

[29] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognit.*, vol. 40, no. 12, pp. 3358–3378, 2007.
[30] L. Bao, J. Cao, J. Li, and Y. Zhang, "Boosted near-miss under-sampling on SVM ensembles for concept detection in large-scale imbalanced datasets," *Neurocomputing*, vol. 172, pp. 198–206, Jan. 2016.
[31] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," in *Proc. Eur. Conf. Principles Data Min. Knowl. Disc.*, 2003, pp. 107–119.
[32] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano, "Rusboost: A hybrid approach to alleviating class imbalance," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 1, pp. 185–197, Jan. 2010.
[33] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *Proc. IEEE Symp. Comput. Intell. Data Min.*, vol. 1, 2009, pp. 324–331.
[34] S. Wang, L. L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1356–1368, May 2015.
[35] D. Tao, X. Tang, X. Li, and X. Wu, "Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 7, pp. 1088–1099, Jul. 2006.
[36] E. Y. Chang, B. Li, G. Wu, and K. Goh, "Statistical learning for effective visual information retrieval," in *Proc. Int. Conf. Image Process.*, 2003, pp. 609–612.
[37] H. Shohei, K. Hisashi, and T. Yutaka, "Roughly balanced bagging for imbalanced data," *Stat. Anal. Data Min.*, vol. 2, no. 2, pp. 412–416, 2010.
[38] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 2, pp. 539–550, Apr. 2009.
[39] O. Pujol and D. Masip, "Geometry-based ensembles: Toward a structural characterization of the classification boundary," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 1140–1146, Jun. 2009.
[40] P. Casale, O. Pujol, and P. Radeva, "Approximate convex hulls family for one-class classification," in *Proc. Int. Conf. Multiple Classifier Syst.*, 2011, pp. 106–115.
[41] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2011.
[42] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.
[43] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.
[44] M. Hollander, D. Wolfe, and E. Chicken, *Nonparametric Statistical Methods*. Hoboken, NJ, USA: Wiley, 2013.

Zonghai Zhu received the B.Sc. degree in information, mechanical, and electrical engineering from Shanghai Normal University, Shanghai, China, in 2010. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai.

His current research interests include pattern recognition and imbalanced problems.

Zhe Wang received the B.Sc. and Ph.D. degrees in computer science and engineering from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2003 and 2008, respectively.

He is currently a Full Professor with the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China. His current research interests include feature extraction, kernel-based methods, image processing, and pattern recognition. He has over 40 papers published in some famous international journals, including the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and *Pattern Recognition*.

Dongdong Li received the B.Sc. and Ph.D. degrees in computer science and engineering from Zhejiang University, Hangzhou, China, in 2003 and 2008, respectively.

She is currently a Full Assistant Professor with the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China. She has over 15 papers published in some famous international journals and conferences. Her current research interests include speech processing, affective computing and pattern recognition.

Yujin Zhu received the B.Sc. and Ph.D. degrees in computer science and engineering from East China University of Science and Technology, Shanghai, China, in 2017.

His current research interests include pattern recognition and machine learning.

Wenli Du received the B.S. and M.S. degrees in chemical process control from the Dalian University of Technology, Dalian, China, in 1997 and 2000, respectively, and the Ph.D. degree in control theory and control engineering from the East China University of Science and Technology, Shanghai, China, in 2005.

She is currently a Professor and the Dean of the College of Information Science and Engineering and the Vice Dean of the Key Laboratory of Advanced Control and Optimization for chemical process, Ministry of Education, East China University of Science and Technology. Her current research interests include control theory and applications, system modeling, advanced control, and process optimization.