Appl Intell (2018) 48:2441–2457
https://doi.org/10.1007/s10489-017-1088-8

A synthetic neighborhood generation based ensemble learning for the imbalanced data classification

Zhi Chen[1] · Tao Lin[1] · Xin Xia[1] · Hongyan Xu[1] · Sha Ding[1]

Published online: 4 December 2017

Abstract Constructing effective classifiers from imbalanced datasets has emerged as one of the main challenges in the data mining community, due to its increased prevalence in various real-world domains. Ensemble solutions are quite often applied in this field for their ability to provide better classification ability than single classifier. However, most existing methods adopt data sampling to train the base classifier on balanced datasets, but not to directly enhance the diversity. Thus, the performance of the final classifier can be limited. This paper suggests a new ensemble learning that can address the class imbalance problem and promote diversity simultaneously. Inspired by the localized generalization error model, this paper generates some synthetic samples located within some local area of the training samples, and trains the base classifiers with the union of original training samples and synthetic neighborhoods samples. By controlling the number of generated samples, the base classifiers can be trained with balanced datasets. Meanwhile, as the generated samples can extend different parts of the original input space and can be quite different from the original training samples, the obtained base classifiers are guaranteed to be accurate and diverse. A thorough experimental study on 36 benchmark datasets was performed, and the experimental results demonstrated that our proposed method can deliver significant better performance than the state-of-the-art ensemble solutions for the imbalanced problems.

✉ Tao Lin
sculintao@gmail.com

[1] College of Computer Science, Sichuan University, Sichuan, China

Keywords Class imbalance problem · Ensemble learning · Synthetic neighborhood generation · Diversity · Classification

1 Introduction

The ultimate goal of constructing a classifier is to learn a set of essential classification rules that can provide accurate classifications on previously unseen data. However, when the class imbalance problem occurs (i.e., the instances of one class significantly outnumber the others), learning essential rules can be challenging. Most of the standard classification algorithms are designed to work on balanced datasets, and they tend to generate classification models maximizing the overall accuracy. Consequently, the resulting classifier will strongly favor the instances of the majority class, because the rules that correctly predict those instances are positively weighted in favor of the overall accuracy, whereas the rules that predict instances of the minority class are fewer and weaker, because minority class is both outnumbered and under presented [1–3]. This could be problematic, because the instances of the minority class usually represent the concept of greater interest, and biased classification on these instances can make the classifier useless during practical use.

Learning from imbalanced datasets has become a critical area in the machine learning community [4] due to its increased prevalence in various real-world domains. Many approaches have been suggested for tackling the class imbalance problem. Data sampling is one of the most commonly used techniques in this field. These methods attempt to artificially rebalance the data distribution by increasing the number of the minority instances (*oversampling*) [5–8],

decreasing the number of the majority instances (*undersampling*) [9–11], or combing them both [12, 13]. Data sampling techniques are classifier-independent solutions to the class imbalance problem, and their general effectiveness has been proven [2, 3]. However, undersampling methods might remove some useful information that could be very important for the classifier training, whereas oversampling methods may increase the possibility of overfitting during classifier training [2]. Cost-sensitive learning (*CSL*) has also been used to address the class imbalance problem. This method assigns different misclassification costs for different classes, generally a higher cost for the minority class and a lower cost for the majority class, and therefore, it attempts to minimize the overall misclassification cost. Although the idea of *CSL* seems to be intuitive, this method is much less popular than data sampling methods. It is difficult to determine a suitable misclassification cost for a given task, and different misclassification costs result in classifiers with different generalization ability. Thus the classification results are not stable [14]. Besides, it presents a significant technical hurdle for those researchers who are not expert in machine learning to modify the learning algorithm for incorporating the idea of *CSL*.

Compared with individual classifiers, ensemble solutions have acquired popularity in this domain for their better performance acquisition [2]. However, to address the class imbalance problem, the ensemble methods need to combine with techniques like oversampling and undersampling [2, 3]. The primary motivation of adopting these techniques is to train the base classifiers on a balanced dataset, not to directly promote the diversity within an ensemble. Thereby, the performance of the final ensemble can be limited. Increasing diversity and decreasing the individual error are two crucial factors for the construction of an optimal ensemble. The general anticipation for an ensemble is that the base classifiers are accurate and mutually complementary so that all the base classifiers can perform like a unified one. Diversity allows different classifiers to offer complementary information for the classification, which in turn, can result into better classification ability [15]. Although, previous study [16] has revealed that diversity-increasing technique can significantly improve the performance of ensemble methods for imbalanced problem, a limited number of studies [16, 17] have been proposed to address the class imbalance problem and promote diversity simultaneously. This study attempts to propose a new ensemble solution that brings these two issues into one unified framework. To accomplish so, a special type of samples, called the neighborhoods of the training samples, is synthetized and imported into the ensemble learning. The generation of these samples is inspired by the localized generalization error model (L-GEM) [18].

L-GEM evaluates the generalization ability of a classifier in a restricted input space. In the authors' opinion, the commonly used learning algorithms, such as SVM and neural network, are local learning machines. The classification boundary of such classifier is shaped by the patterns hidden in the training samples, and samples far away from the training samples will not affect the construction of classification boundary. So, instead of evaluating a classifier's classification ability in the entire input space. Yeung et al. [18] proposed a L-GEM to assess the classification ability of radial basis function neural networks (RBFNNs) in some limited neighborhoods of the training samples. The idea proves to be effective and efficient in many applications, such as model selection [19] and feature selection [20] for RBFNNs.

Our previous study [21] has incorporated the synthetic neighborhoods into an ensemble learning to maximize the overall accuracy and achieved a significant improvement in generalization ability for the balanced datasets. In this paper, we propose a sample-generation-based ensemble learning for the class imbalance problem. When training a base classifier, the proposed method randomly selects a subset of training samples from the original training set, and replaces them with their synthetic neighborhoods. The synthetic neighborhoods are beneficial extensions to the input space of original training samples, so the base classifiers can improve their classification ability in different parts of the input space by learning these synthetic samples. Moreover, as the generated samples are randomly generated and can be different from the original training samples, the diversity within the final ensemble can be promoted. In addition, the synthetic samples are different but not conflicting with the original training samples, so we can rebalance the class distribution by generating certain number of minority samples and not worrying about decreasing the base classifiers' generalization ability.

Although L-GEM has highlighted the importance of the neighborhoods of the training samples, it does not generate any actual sample. It is essential to generate available synthetic neighborhoods that can be used to solve the class imbalance problem and promote diversity simultaneously. We provide our solution of synthetic neighborhood generation, and assess its effectiveness on 36 benchmark datasets. The experimental results demonstrated that our proposed method significantly outperformed the state-of-art ensemble learning in terms of area under receiver operation characteristic (AUC) and *G-mean*.

2 Related work

In this paper, the binary-class imbalanced datasets are the focus, in which there is a positive (minority) class, with the

lowest number of instances, and a negative (majority) class, with a significantly higher number of instances. Of the two classes, the minority class is usually the class of interest, but it is difficult to identify because it might be associated with exceptional and significant cases, or the process of acquiring these examples is costly [22]. Over the years, a significant amount of work has been done to address class imbalance problem. These methods can be broadly divided into four categories: (1) algorithm level approaches, (2) data level approaches, (3) cost-sensitive learning, and (4) ensemble solutions. In this section, we only focus on the ensemble solutions that have been proposed to address this issue. A general introduction and additional details about the other techniques can be found in [1–3].

Specifically, the best ensemble solutions to the class imbalance problem are reviewed firstly. Then the commonly used assessment metrics for the imbalanced learning are given.

2.1 Ensemble solutions for the imbalanced learning

Ensembles of classifiers have been adopted and their robustness at handling imbalanced datasets has been proven [10, 16]. Ensemble techniques by themselves do not ameliorate the class imbalance problem, because they are designed to achieve a maximal accuracy on the training set. However, their combination with previous techniques leads to promising results. Ensemble techniques in this field can be classified into two types: cost-sensitive boosting approaches [23] and ensemble learning algorithms with embedded data preprocessing techniques [9, 24–26]. The cost-sensitive boosting approaches share a similar spirit with non-ensemble cost-sensitive approaches, which assign different misclassification cost for different classes. The main difference is that the costs minimization is guided by the boosting algorithm. Cost-sensitive boosting approaches share same idea, as well as same shortcoming, as the non-ensemble cost-sensitive methods: it is not an intuitive task to assign appropriate costs for the different classes. On the other hand, ensemble methods combined with data sampling are more popular. According to Galar et al. [2], these methods can be further classified into three subclasses: (1) bagging-, (2) boosting-, and (3) hybrid-based approaches, depending on the ensemble learning algorithm that they use. These approaches do not change the original process of ensemble learning, they just adopt data-level approaches, such as oversampling and undersampling, to rebalance the data distribution and train the base classifiers with balanced datasets. As most related works [2, 16, 17] indicate good performance of Bagging [27] and Boosting [28] in combination with data-level approaches some of these ensemble algorithms are recalled.

- **SMOTEBagging** [24] has an operating procedure that is similar to that of Bagging, except that each base classifier is trained on a balanced dataset. At each iteration, the method generates a dataset that has two times the number of majority samples. Half of the instances are randomly sampled from the majority class with replacement whereas the second half is generated through a combination of SMOTE [5] and random oversampling on the minority class. The oversampling percentage varies from 10% in the first iteration to 100% in the last, always being a multiple of ten. The remaining minority instances are generated by SMOTE.
- **UnderBagging** [26]. In each iteration of UnderBagging, the number of majority class instances is randomly reduced to the number of the minority class to allow the base classifier to be trained with a balanced dataset.
- **SMOTEBoost** [29] is a modification of AdaBoost.M2 [30]. After each iteration, this approach uses SMOTE to balance the dataset. The newly generated instances will be assigned a weight, which is the proportion of generated instances to the overall number of instances. Other than the newly generated instances, the weights of the original instances will be normalized. During the whole procedure, the instances' weights are updated according to the algorithm of Adaboost.M2.
- **RUSBoost** [25] has a similar procedure to SMOTEBoost. However, it adopts a random undersampling method to balance the dataset. Thus, no instances will be generated, and the weights of the instances will be updated according to the algorithm of Adaboost.M2.
- **EUSBoost** [10] is a modification of RUSBoost. This approach aims to improve the original method by using an evolutionary undersampling approach. To be specific, it attempts to promote diversity by selecting different subsets of the majority instances for different base classifiers with an evolutionary undersampling approach.
- **EasyEnsemble** [9] has a similar procedure to UnderBagging. However, instead of training a base classifier for each new bag, it adopts an AdaBoost as a base classifier. When training each of the base classifiers, EasyEnsemble samples several subsets from the majority class, and trains a base classifier with these subsets repeatedly. So, EasyEnsemble works like an ensemble of ensembles.

Other than combining ensemble with data-level approaches, researchers in this field have been constantly seeking new framework of ensemble learning. For example, the study in [16] demonstrated that diversity-increasing techniques are effective methods for improving

the performance on imbalanced problems. In addition, the study also suggested that diversity-enhancing techniques should be adopted when the overlap of the per-class bounding boxes is high. Bhowan et al. [31] adopted genetic algorithm to evolve ensemble. In their proposed approach, the construction of an ensemble was transformed to a multi-objective optimization problem, in which a set of accurate and diverse base classifiers were built by trading-off the minority and majority class during the learning of base classifier. Díez-Pastor et al. [32] proposed a data-level approach that can be used to build ensembles for class imbalance problem. In their approach, the data for training a base classifier is sampled from the training data using random class proportions. The diversity can be promoted by randomly undersampling or oversampling the original training data.

3 Proposed method

3.1 Motivation

We attempt to propose a new sample generation based ensemble learning for the class imbalance problem, which can bring the issues of addressing the imbalance problem and promoting diversity into one unified framework. Sample generation is not a new idea for ensemble learning. Importing properly synthesized samples into ensemble learning has proven to be an effective way for improving a classifier's generalization ability [33–36]. In previous studies, the synthetic samples are generated mainly for two purposes: (1) addressing the class imbalance problem by generating some new synthetic minority samples (*oversampling*); (2) creating diversity in the ensembles. When applying oversampling techniques to ensemble learning, they are used to address the problem of class imbalance, not to directly enhance the diversity within an ensemble. Thereby, the diversity and generalization ability of the final classifier could be limited.

Sample generation methods [33–36] have been constantly used to increase diversity within an ensemble. These methods generate some synthetic samples and a particular base classifier is trained with the synthetic samples along with the original training samples. Theoretical and empirical studies [37, 38] reveal that one may get very promising diversity by generating very different samples with any of the existing sample generation methods, however, when addressing the imbalance problem, these methods are limited, because: (1) the diversity of the final ensemble is promoted at the cost of base classifiers' classification ability; (2) existing sample generation based ensembles are accuracy-oriented. They are designed to maximize the overall accuracy. Although theoretically, the generated samples can be used to rebalance the class distribution, but importing too many of these samples could be problematic, because the samples generated by any of the existing methods can be quite different from the original training samples [38]. Learning too many these synthetic samples can misguide the training of a classifier.

The work of Yeung et al. [18] highlights the importance of unseen samples located within some neighborhood of the training samples. The effectiveness of L-GEM in various applications [19, 20, 39] also proves that the synthetic samples located within this area can be beneficial extensions to the original input space, and it would be desirable to design a particular ensemble learning, which incorporates these synthetic neighborhoods for the class imbalance problems because we can bring the issues of increasing diversity, decreasing individual error and addressing class imbalance problem into one unified framework, by incorporating the synthetic neighborhoods into the ensemble learning.

3.2 Synthetic neighborhoods generation

It is essential to generate available synthetic neighborhoods that are properly located within some neighboring area of the training samples. The generated samples must have appropriate distance to the training samples, so that a balance between increasing diversity and decreasing individual error can be achieved.

Central Limit Theorem *Let X_i $(i = 1, 2, \ldots, n)$ be a random sample from the same underlying distribution with mean μ and variance σ^2. Let $\overline{X}$ be the average of X_i $(i = 1, 2, \ldots, n)$, if n is "sufficiently large", then $\overline{X}$ obeys a normal distribution with mean μ and variance σ^2.*

$$\frac{\overline{X} - \mu}{\sigma/\sqrt{n}} = \frac{\sum_i^n X_i - n\mu}{\sqrt{n}\sigma} \rightarrow N(0, 1) \tag{1}$$

The central limit theorem shown in (1) can be applied for any distribution. This permits us to generate synthetic neighborhoods for any unknown data distribution.

The symbols that will be used in this paper are described as follows. $D = \{(x_i, y_i)|x_i \in \mathbb{R}^p, y_i \in \mathcal{Y}\}_{i=1}^N$ is the training set, where p is the number of input features, N denotes the number of training samples, and $\mathcal{Y} = \{label_1, label_2\}$ indicates the class labels in a binary classification problem.

In a multiple feature dataset D, each training sample is composed of multiple features, and each feature can be treated as a random variable that obeys an unknown distribution. When generating one synthetic neighborhood for a specific training sample, a new dataset $D_j = \{(x_i, y_i)|(x_i, y_i) \in D, y_i = label_j)\}$ is gathered by selecting all the samples labeled with $label_j$ in D. After that, the mean value μ_i and standard variance σ_i for the ith attribute a_i in D_j are calculated, and we use μ_i' and σ_i' to represent

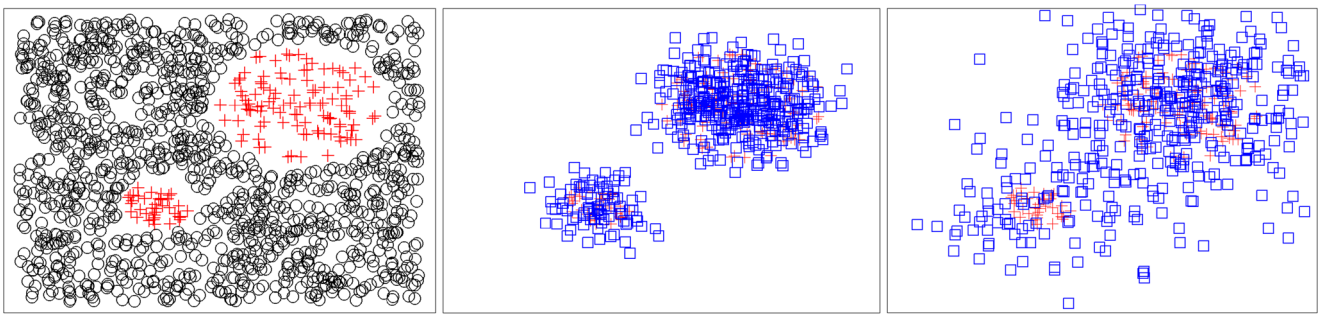

Fig. 1 **a** Original simulated dataset. **b** The synthetic minority instances (blue hollow squares) generated by our method when $\lambda=1/\sqrt{n}$. (c) $\lambda = 0.5$

the real mean and standard deviation of attribute a_i. n is the number of samples in D_j. If n is large enough, according to Central Limit Theorem, the following can be obtained:

$$\frac{\mu - \mu_i'}{\sigma_i'/\sqrt{n}} \rightarrow N(0,1) \tag{2}$$

Because (2) is satisfied only when n is "sufficiently large", it can be used to approximate the real mean value μ_i':

$$\mu_i' = \mu - r \times \frac{\sigma_i'}{\sqrt{n}} \tag{3}$$

Where r is a sampling value of normal distribution $N(0,1)$. By substituting the mean value μ with an feature value, then for each instance j in D_j, given its ith attribute value $a_i(j)$, a synthetic value $a_i'(j)$ can be generated as follows:

$$a_i'(j) = a_i(j) - r \times \frac{\sigma_i'}{\sqrt{n}} \tag{4}$$

Here, σ_i' in (4) represents the true standard deviation of the attribute a_i in D_j, and it is unknown. We can approximate it with σ_i, and the following can be obtained:

$$a_i'(j) = a_i(j) - r \times \frac{\sigma_i}{\sqrt{n}} \tag{5}$$

Zhang and Li [40] use the equation (5) as an oversampling technique. The samples generated by oversampling techniques are safely located within the area that surrounds all the training samples. The method combining ensemble learning with synthetic samples generated by (5) has no essential difference to current ensemble solutions for imbalanced dataset. So, the size of the synthetic attribute is expanded, so that the diversity within final ensemble can be enhanced

$$a_i'(j) = a_i(j) - \lambda r \sigma_i \tag{6}$$

Equation (6) is the one used to generate the synthetic neighborhoods, where parameter λ is a trade-off between

Fig. 2 Pseudo code of the proposed ensemble method

Input:

- D: The training set, $D = \{(x_i, y_i) \mid x_i \in \mathbb{R}^p, y_i \in \mathcal{Y}\}_{i=1}^N$
- M: The number of base classifiers.
- RR: The parameter determining the proportion of samples that needs to be replaced.
- λ : The parameter used to control the distance from the synthetic samples to the training samples.

Training Begin:

- for i=1 to M
  - get a copy of the original training set, $D_i = D$
  - get the number of training samples that will be replaced as follows: $P = round(n \times RR)$
  - for j=1 to P
    - Randomly select a sample $\boldsymbol{x}$ from D_i
    - **If** $\boldsymbol{x}$ is a majority class sample, then
      - → Generate a synthetic neighborhood of $\boldsymbol{x}$ according to equation (6), and replace $\boldsymbol{x}$ in D_i with the synthetic neighborhood
    - **Elseif** $\boldsymbol{x}$ is a minority sample, then
      - → Generate m synthetic neighborhoods of $\boldsymbol{x}$ according to equation (6), and replace $\boldsymbol{x}$ in D_i with the m synthetic neighborhoods. m can be calculate by equation (7)
  - End loop
  - Construct a base classifier C_i from D_i.
- End loop

Classification Phase:

- For a given $\boldsymbol{x}$, use the ensemble $\{C_1, C_2, ..., C_M\}$ to classify the sample $\boldsymbol{x}$ with simple majority voting strategy

Table 1 Dataset details

No.	Name	IR	#Ex	#Atts
1	ecoli0vs1	1.86	220	7
2	pimaimb	1.87	768	8
3	iris0	2.0	150	4
4	glass0	2.06	214	9
5	yeast1	2.46	1484	8
6	vehicle2	2.88	846	18
7	vehicle3	2.99	846	18
8	glass0123vs456	3.20	214	9
9	vehicle0	3.25	846	18
10	ecoli1	3.36	336	7
11	newthyroid2	5.14	215	5
12	newthyroid1	5.14	215	5
13	ecoli2	5.46	336	7
14	segment0	6.02	2308	19
15	glass6	6.38	214	9
16	yeast3	8.10	1484	8
17	ecoli3	8.6	336	7
18	yeast2vs4	9.08	514	8
19	vowel0	9.98	988	13
20	glass016vs2	10.29	192	9
21	glass2	11.59	214	9
22	shutlec0vsc4	13.87	1829	9
23	glass4	15.46	214	9
24	ecoli4	15.8	336	7
25	abalone9-18	16.40	731	8
26	glass016vs5	19.44	184	9
27	shuttlec2vsc4	20.5	129	9
28	glass5	22.78	214	9
29	yeast5	32.72	1484	8
30	ecoli0137vs26	39.14	281	7
31	abalone17vs78910	39.31	2338	8
32	krvskvs8	53.07	1460	6
33	shuttle2vs5	66.67	3316	9
34	kddcupland*vs*satan	75.67	1610	41
35	kr*vs*k*vs*15	80.22	2193	6
36	kddcuprootkitimap*vs*back	100.13	2225	41

Table 3 Confusion matrix

	Predicted positive	Predicted negative
Actual positive	TP	FN
Actual negative	FP	TN

increasing diversity and decreasing individual error. A larger λ will result into a set of synthetic neighborhoods that are very different to the corresponding training samples. The opposite scenario can happen if we decrease λ. It is also worth mentioning that the equation (6) considers some kind of global distribution information by importing the standard deviation σ_i into the equation. By doing this, the proposed method can work reasonably well under some fixed settings of parameter λ.

For each sample in $D_j = \{(x_i, y_i)|(x_i, y_i) \in D, y_i = label_j)\}$, we use (6) to generate one synthetic feature value for each of its attribute, and a feature vector can be constructed by gathering all the corresponding synthetic values. $label_j$ will be assigned to the feature vector as its class label, and then the vector is taken as a synthetic neighborhood to the training sample. In order to illustrate the proposed method in better way, we run our sample generation method on a simulated dataset. Figure 1 shows the results. Figure 1a presents the simulated dataset, where the black circle points denote the majority class instances and the red plus symbols denote the minority class instance. The blue hollow square symbols in Fig. 1b and c denote the synthetic minority instances generated by (6) under different settings of parameter λ.

3.3 Synthetic neighborhood generation based ensemble learning for imbalanced problem (SNGEIP)

We formally propose our ensemble solution for imbalanced problem (SNGEIP) in Fig. 2. The basic aim of the sample generation is to create different training sets for different base classifiers so that a promising diversity can be

Table 2 State-of-the-art ensembles used in our experiments

Abbr.	Method	Description
EUS	EUSBoost [10]	Adaboost embedded with evolutionary random undersampling
SBO	SMOTEBoost [29]	RUS embedded with SMOTE
RUS	RUSBoost [25]	Adaboost embedded with random undersampling
UBAG	UnderBagging [26]	Bagging embedded with undersampling
SBGA	SMOTEBagging [24]	Bagging embedded with SMOTE
SESNG	SMOTE + ESNG [21]	ESNG method combined with SMOTE
EASY	EasyEnsemble [9]	Bagging with undersampling of the majority class and Adaboost

achieved. At the same time, the class distribution can be rebalanced by controlling the number of generated samples.

Specifically, given a training set $D = \{(x_i, y_i)\}_{i=1}^{n}$, where n is the number of samples in D, and IR denotes the imbalance ratio of D. The proposed SNGEIP method first gets a copy of the original training set, and randomly selects some training samples. The selected samples will be replaced by their synthetic neighborhoods. If a majority sample is chosen, it will be replaced by its one synthetic neighborhood generated by (6). Otherwise, if a minority sample is chosen, it will be replaced by its m synthetic neighborhoods. The synthetic minority samples are more than the majority samples, so the class distribution can be rebalanced. m can be calculated as follows:

$$m = Round((IR - 1)/RR + 1) \tag{7}$$

Where IR is the imbalance ratio of D, and RR is the abbreviation for the replacement ratio, a parameter used to determine the proportion of training samples that needs to be replaced. RR can be set as any value in the range [0, 1]. If we set $RR = 0$, no synthetic sample will be generated and we will obtain a set of identical base classifiers. If we set $RR = 1$, every training sample must be replaced by their synthetic neighborhoods and the diversity within the final ensemble can be promoted. The choices of RR and λ have significant influences on the generalization ability of final ensemble. In this paper, the default settings for these two parameters are $RR = 0.368$, and $\lambda = 0.5$. We give the reason for this default setting in Section 4.6, where we can discuss the influence of these two parameters.

4 Experiments

4.1 Benchmark datasets

A series of experiments were conducted to test the performance of our method, the synthetic neighborhood generation based ensemble learning for imbalanced problem (SNGEIP). We have selected 36 imbalanced binary datasets from the KEEL repository [41], and the detailed information about these datasets is given in Table 1. The datasets are ordered according to their IRs. Before testing the performance of our method, all the datasets need to be preprocessed. All the datasets have been transformed into Libsvm format [42], a data format that only contains numeric type attributes. By doing this, we can automatically transform all the non-numeric attributes into numeric one, and calculate the standard deviation for each attribute.

4.2 Benchmark algorithms and parameter settings

We analyze the quality of our proposed method against seven other methods (described in Section 2). To be specific, we only compare our proposed method with Bagging

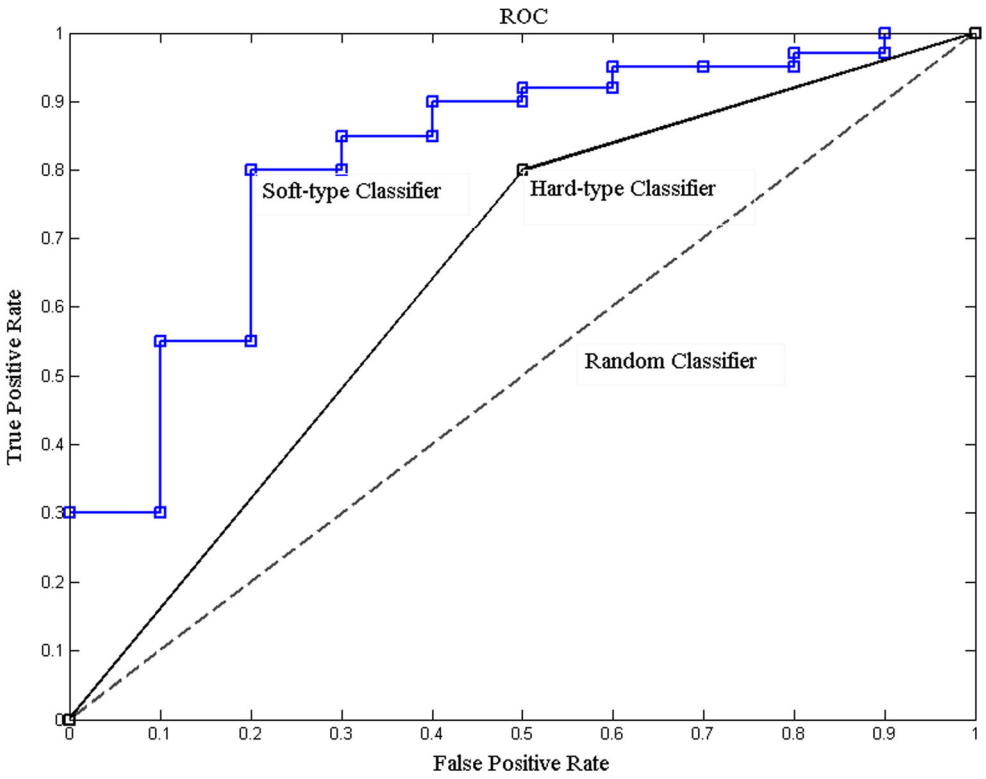

Fig. 3 Example of ROC curve. Three classifiers' curves are plotted. The area under the curve is the AUC

Table 4 Average AUCs obtained from 3 × 5 cross validation for all the ensembles

Datasets	IR	Boosting-based			Bagging-based			Hybrid	
		EUS	SBO	RUS	UBAG	SBAG	SENSG	EASY	SNGEIP
ecoli0vs1	1.86	0.9819 ± 0.07	0.9896 ± 0.05	0.9830 ± 0.07	0.9854 ± 0.06	0.9731 ± 0.13	0.9858 ± 0.02	0.9776 ± 0.09	**0.9964 ± 0.02**
pimaimb	1.87	0.8182 ± 0.09	0.8176 ± 0.10	0.8095 ± 0.11	0.8299 ± 0.11	0.8334 ± 0.11	0.8276 ± 0.04	0.7569 ± 0.13	**0.8356 ± 0.11**
iris0	2.0	0.9833 ± 0.14	0.9833 ± 0.14	0.9833 ± 0.14	0.9833 ± 0.14	0.9733 ± 0.16	0.9867 ± 0.03	0.9833 ± 0.14	**0.9867 ± 0.11**
glass0	2.06	0.9005 ± 0.22	**0.9275 ± 0.12**	0.9100 ± 0.13	0.9053 ± 0.14	0.9056 ± 0.13	0.8787 ± 0.05	0.8469 ± 0.30	0.9133 ± 0.14
yeast1	2.46	0.7805 ± 0.08	0.7858 ± 0.12	0.7828 ± 0.10	0.7952 ± 0.11	**0.7962 ± 0.10**	0.7884 ± 0.02	0.7512 ± 0.14	0.7882 ± 0.10
vehicle2	2.88	0.9957 ± 0.01	**0.9988 ± 0.01**	0.9956 ± 0.01	0.9954 ± 0.01	0.9965 ± 0.01	0.9941 ± 0.01	0.9831 ± 0.03	0.9967 ± 0.01
vehicle3	2.99	0.8582 ± 0.09	0.8522 ± 0.12	0.8568 ± 0.15	0.8574 ± 0.13	0.8515 ± 0.11	**0.8587 ± 0.02**	0.7675 ± 0.19	**0.8587 ± 0.12**
glass0123vs456	3.20	0.9629 ± 0.12	0.9784 ± 0.08	0.9696 ± 0.12	0.9713 ± 0.13	0.9767 ± 0.10	0.9661 ± 0.03	0.9512 ± 0.15	**0.9792 ± 0.09**
vehicle0	3.25	0.9906 ± 0.01	**0.9962 ± 0.01**	0.9922 ± 0.02	0.9899 ± 0.03	0.9918 ± 0.02	0.9904 ± 0.01	0.9725 ± 0.03	0.9917 ± 0.02
ecoli1	3.36	0.9514 ± 0.12	0.9631 ± 0.07	0.9538 ± 0.09	0.9554 ± 0.11	**0.9636 ± 0.08**	0.9475 ± 0.03	0.9157 ± 0.14	0.9616 ± 0.07
newthyroid2	5.14	0.9843 ± 0.06	**0.9971 ± 0.02**	0.9787 ± 0.11	0.9618 ± 0.19	0.9958 ± 0.02	0.9820 ± 0.04	0.9611 ± 0.18	0.9937 ± 0.05
newthyroid1	5.14	0.9729 ± 0.14	**0.9992 ± 0.01**	0.9766 ± 0.12	0.9690 ± 0.13	0.9947 ± 0.03	0.9840 ± 0.04	0.9758 ± 0.11	0.9989 ± 0.01
ecoli2	5.46	0.9477 ± 0.11	**0.9696 ± 0.10**	0.9541 ± 0.16	0.9406 ± 0.17	0.9522 ± 0.16	0.9470 ± 0.04	0.9022 ± 0.19	0.9649 ± 0.12
segment0	6.02	0.9976 ± 0.02	0.9959 ± 0.04	0.9978 ± 0.01	0.9940 ± 0.03	0.9973 ± 0.03	0.9981 ± 0.00	0.9933 ± 0.03	**0.9997 ± 0.00**
glass6	6.38	0.9383 ± 0.25	**0.9649 ± 0.15**	0.9326 ± 0.25	0.9339 ± 0.24	0.9619 ± 0.17	0.9420 ± 0.06	0.9196 ± 0.27	0.9573 ± 0.21
yeast3	8.10	0.9683 ± 0.06	0.9677 ± 0.04	0.9661 ± 0.05	0.9758 ± 0.04	**0.9765 ± 0.03**	0.9713 ± 0.01	0.9385 ± 0.08	0.9747 ± 0.04
ecoli3	8.6	0.9183 ± 0.16	0.9464 ± 0.12	0.9166 ± 0.19	0.9356 ± 0.16	0.9309 ± 0.15	0.9043 ± 0.08	0.9091 ± 0.19	**0.9470 ± 0.15**
yeast2vs4	9.08	0.9700 ± 0.09	0.9758 ± 0.06	0.9547 ± 0.24	0.9536 ± 0.23	0.9822 ± 0.04	0.9764 ± 0.02	0.9372 ± 0.22	**0.9839 ± 0.04**
vowel0	9.98	0.9949 ± 0.02	0.9983 ± 0.01	0.9897 ± 0.04	0.9849 ± 0.05	0.9987 ± 0.01	0.9987 ± 0.00	0.9805 ± 0.06	**0.9998 ± 0.00**
glass016vs2	10.29	0.7999 ± 0.35	0.8333 ± 0.37	0.7957 ± 0.44	0.7744 ± 0.44	0.8417 ± 0.32	0.7891 ± 0.13	0.7004 ± 0.53	**0.8532 ± 0.34**
glass2	11.59	0.7572 ± 0.47	0.8723 ± 0.43	0.7848 ± 0.45	0.8130 ± 0.34	0.8777 ± 0.36	0.8965 ± 0.06	0.6904 ± 0.63	**0.8806 ± 0.35**
shutlec0vsc4	13.87	**1.0000 ± 0.00**	0.9997 ± 0.00	**1.0000 ± 0.00**	**1.0000 ± 0.00**	0.9996 ± 0.00	**1.0000 ± 0.00**	**1.0000 ± 0.00**	**1.0000 ± 0.00**
glass4	15.46	0.8533 ± 0.61	0.9710 ± 0.13	0.8866 ± 0.47	0.8987 ± 0.46	**0.9779 ± 0.08**	0.9471 ± 0.08	0.8943 ± 0.56	0.9778 ± 0.14
ecoli4	15.8	0.9549 ± 0.16	0.9908 ± 0.03	0.9398 ± 0.22	0.9287 ± 0.25	0.9906 ± 0.03	0.9379 ± 0.08	0.9196 ± 0.22	**0.9908 ± 0.03**
abalone9-18	16.40	0.8006 ± 0.30	0.8131 ± 0.29	0.8171 ± 0.33	0.8264 ± 0.28	0.8044 ± 0.23	0.8027 ± 0.10	0.7240 ± 0.46	**0.8461 ± 0.25**
glass016vs5	19.44	0.9829 ± 0.06	**0.9948 ± 0.04**	0.9271 ± 0.32	0.9300 ± 0.32	0.9943 ± 0.04	0.9943 ± 0.01	0.9348 ± 0.32	0.9914 ± 0.06
shuttlec2vsc4	20.5	0.9944 ± 0.08	**1.0000 ± 0.00**	0.9611 ± 0.34	0.9611 ± 0.34	**1.0000 ± 0.00**	**1.0000 ± 0.00**	0.9611 ± 0.34	**1.0000 ± 0.00**
glass5	22.78	0.9740 ± 0.10	0.9866 ± 0.11	0.9484 ± 0.13	0.9455 ± 0.12	0.9935 ± 0.07	**0.9980 ± 0.00**	0.9480 ± 0.13	0.9967 ± 0.03
yeast5	32.72	0.9783 ± 0.10	0.9893 ± 0.03	0.9811 ± 0.11	0.9793 ± 0.11	0.9912 ± 0.02	0.9929 ± 0.00	0.9672 ± 0.11	**0.9936 ± 0.01**
ecoli0137vs26	39.14	0.8250 ± 0.79	0.8827 ± 0.71	0.8176 ± 0.80	0.8267 ± 0.57	0.8451 ± 0.97	0.8342 ± 0.22	0.8385 ± 0.60	**0.9370 ± 0.51**

Table 4 (continued)

Datasets	IR	Boosting-based			Bagging-based			Hybrid	
		EUS	SBO	RUS	UBAG	SBAG	SENSG	EASY	SNGEIP
abalone17vs78910	39.31	0.8647 ± 0.25	0.9167 ± 0.10	0.8850 ± 0.21	0.9092 ± 0.17	**0.9284 ± 0.09**	0.9238 ± 0.03	0.8454 ± 0.21	0.9256 ± 0.10
krvskvs8	53.07	0.9845 ± 0.07	0.9999 ± 0.00	0.9709 ± 0.14	0.9660 ± 0.13	0.9994 ± 0.01	0.9996 ± 0.00	0.9644 ± 0.13	**1.0000 ± 0.00**
shuttle2vs5	66.67	0.9969 ± 0.04	**1.0000 ± 0.00**	0.9970 ± 0.04	0.9970 ± 0.04	**1.0000 ± 0.00**	**1.0000 ± 0.00**	0.9970 ± 0.04	**1.0000 ± 0.00**
kddcupplandvssatan	75.67	1.0000 ± 0.00	**1.0000 ± 0.00**	**1.0000 ± 0.00**	**1.0000 ± 0.00**	**1.0000 ± 0.00**	**1.0000 ± 0.00**	**1.0000 ± 0.00**	**1.0000 ± 0.00**
krvskvs15	80.22	0.9834 ± 0.07	**1.0000 ± 0.00**	0.9947 ± 0.04	0.9912 ± 0.04	**1.0000 ± 0.00**	**1.0000 ± 0.00**	0.9956 ± 0.04	0.9999 ± 0.00
kddcuprootkitimapvsback	100.13	0.9745 ± 0.28	**0.9998 ± 0.00**	0.9818 ± 0.26	0.9818 ± 0.26	0.9998 ± 0.00	**1.0000 ± 00.0**	0.9818 ± 0.26	0.9917 ± 0.12
Average		0.9345	0.9544	0.9331	0.9346	0.9527	0.9457	0.9107	**0.9587**

and Boosting in combination with data-level approaches. After comparing 20 different ensembles from simple modifications of Bagging or Boosting to complex cost or hybrid approaches, Galar et al. [2] found that methods combining Bagging or Boosting with simple versions of undersampling or SMOTE work better than more complex ensemble solutions. Other than Bagging and Boosting methods, we combined ESNG [21] method with SMOTE method. ESNG [21] incorporated the synthetic neighborhoods into an ensemble learning to maximize the overall accuracy and achieved a significant improvement in generalization ability for the balanced datasets. We combined it with SMOTE method and tested its performance on imbalanced datasets. It will better reveal the advantages or drawbacks of our method by comparing it with ESNG method. Table 2 shows the operating procedure and the abbreviations that will be used through the experiments. The C4.5 decision tree was chosen as the base classifier for all the ensemble learning.

40 decision trees were trained for each of the ensemble methods. It is worth mentioning that the choices of parameters RR and λ have significant influence on the classification ability of our proposed method. In all the experiments, we set $RR = 0.368$, and $\lambda = 0.5$. All experiments have been conducted using the KEEL software [41] and Weka [43]. The ESNG combined with SMOTE method has been implemented in Weka, whereas the other six learning algorithms in Table 2 are publicly available in KEEL.

4.3 Assessment metrics used in the experiments

The way of assessing a classifier is very important for properly evaluating its classification performance and guiding its modeling. In order to take the class distribution into account, we must use specific metrics to evaluate the classifier's performance. Focusing on binary-class problems, the confusion matrix (shown in Table 3) records the results of correctly and incorrectly recognized examples of each class. Specifically, *TP* represents the number of positive examples that have been correctly predicted as positive, *FN* represents the positive example number that have been falsely predicted as negative, *FP* represents the negative example number that have been falsely predicted as positive, and *TN* represents the negative example number that have been correctly predicted as negative. From Table 3, some metrics can be calculated and used as assessment metrics in the imbalance framework:

True positive rate: $$TP_{rate} = TP/(TP + FN) \quad (8)$$

True negative rate: $$TN_{rate} = TN/(FP + TN) \quad (9)$$

Precision: $$P = TP/(TP + FP) \quad (10)$$

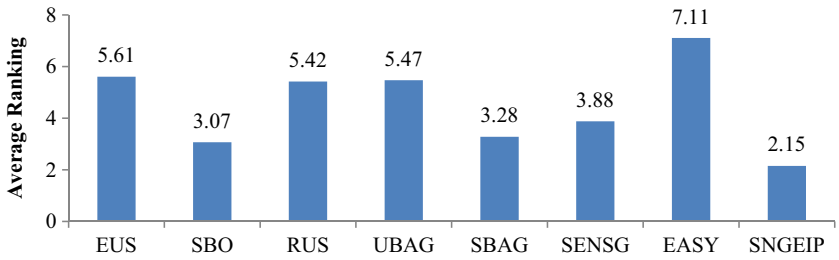

Fig. 4 Average rankings obtained from Friedman aligned-rank test based on the AUCs

$$\textit{Recall:} \quad R = TP/(TP + FN) \tag{11}$$

$$F\text{-measure:} \quad F = (2 \times P \times R)/(P + R) \tag{12}$$

$$\textit{G-mean:} \quad Gmean = \sqrt{TP_{rate} \times TN_{rate}} \tag{13}$$

Clearly, the first four measures in (8–11) describe classifier's performance on positive class and negative class separately, but none of these measures is adequate enough to describe the overall performance. Both F-measure and *G-mean* can be used as the measures to evaluate a classifier's performance in class imbalance problem. But when a classifier classifies all the instances as negative class (i.e., $TP = 0$, and $FP = 0$), the F-measure would be $+\infty$, making the experimental results incomparable.

Another well-known approach to produce an overall evaluation criterion is the receiver operation characteristic (ROC) curve [44]. ROC curve is formed by plotting TP_{rate} over FP_{rate}, and any point in ROC space corresponds to the performance of a single classifier on a given distribution. ROC curve provides a visual representation of the trade-off between benefits (TP_{rate}) and costs (FP_{rate}) of classification. Area under the ROC curve (AUC) [45] provides a quantitative measure of a classifier's performance for the evaluation of which model is better. The computation of AUC depends on classifier's type. If the classier is hard-type which outputs only discrete class labels, the AUC can be computed as:

$$AUC = (1 + TP_{rate} - FP_{rate})/2 \tag{14}$$

If the classifier is soft-type which outputs a continuous numeric value to represent the confidence of an example belonging to the predicted class, a threshold can be used to produce a series of points in ROC space. The AUC is computed as the area under these points. Figure 3 shows the idea of ROC and AUC.

In this paper, *G-mean* and AUC are adopted to evaluate the classifier's performance. Considering the fact that all the classifiers used in our experiment are soft-type classifiers, we used a threshold to produce a series of points in ROC space, and the AUC is computed as the area under these points.

4.4 Experimental procedure and statistical tests

We follow a procedure that is similar to that outlined in [10]. Specifically, an experiment framework using 3×5 cross-validation is adopted to compare the performance of different algorithms. For each dataset, 5-fold stratified cross validation is used to divide the dataset into training parts and testing parts. The training parts are used to train the classifiers, while the testing parts are used to calculate the AUC and *G-mean* metrics. Here, 5-fold cross validation was conducted three times with different random seeds, and the average value of all the obtained AUCs or *G-means* can be used as a single measure, which provides a reliable estimation on the classifier's ability of dealing with imbalanced datasets.

Statistical analysis needs to be conducted in order to determine whether the classification ability of different

Table 5 Holm post-hoc test results based on the AUCs

Control method (SNGEIP)

	Algorithm (ranking)	Z	p-value	Holm	Hypothesis ($\alpha = 0.05$)
7	EASY(7.11)	8.5881	0.0000	0.0071	**Rejected**
6	EUS(5.61)	5.9900	0.0000	0.0083	**Rejected**
5	UBAG(5.47)	5.7494	0.0000	0.0100	**Rejected**
4	RUS (5.42)	5.6532	0.000	0.0125	**Rejected**
3	SENSG (3.88)	3.0070	0.0043	0.0167	**Rejected**
2	SBAG(3.28)	1.9486	0.0598	0.0250	Not rejected
1	SBO(3.07)	1.5877	0.1131	0.0500	Not rejected

The bold emphasis indicates the significant statistical test results

Table 6 Wilcoxon pairwise test result of SNGEIP vs. SBAG and SBO based on the AUCs

Comparison	R+	R−	p-value	Hypothesis (0.05)
SNGEIP vs. SBAG	414.0	147.0	0.017	**Rejected**
SNGEIP vs. SBO	397.5	163.5	0.037	**Rejected**

The bold emphasis indicates the significant statistical test results

algorithms are significantly different. As suggested by previous study [46], we consider two different non-parametric statistical tests to perform two comparisons:

- **Multiple comparisons**. As suggested by Demšar [46], we first use Friedman test [47] with its corresponding post-hoc test to detect statistical differences between all the methods. Then, if significant differences exist, we check whether the control method (the one with lowest average ranking) is significantly better than the others using Holm test [48].
- **Pairwise comparisons**. Pairwise comparison is used as a complementary test here, because the Friedman test occasionally reports a significant difference but the post-hoc test might fail to detect it. Wilcoxon paired signed-rank test was conducted to further confirm whether the classification abilities of two methods are significantly different.

Moreover, instead of simply giving an overall summary, we show the p-value associated with each comparison, which indicates the lowest level of significance of a hypothesis that results in a rejection. In such a manner, we can know whether two algorithms are significantly different and how different they are. The average rankings of each method have also been used as a complementary visualization tool. These rankings are obtained from the Friedman aligned-rank test, and a lower ranking indicates better classification ability.

4.5 Performance of SNGEIP and comparison with conventional methods

4.5.1 AUC as a measure

Table 4 shows the average AUC over 3 × 5 cross validation in the form of 'average ± standard deviation'. For each dataset, the best AUC over eight methods is shown in boldface type. Taking a quick glance at this table shows that the proposed work is the best performing method, because it achieves the highest number of best AUCs (on 20 of the 36 datasets) and the highest average AUC (0.9587) over all the eight methods. In order to check if the higher average AUC indicates actual better classification ability, non-parametric statistical tests have been conducted on the obtained AUCs. The average rankings from the Friedman test are shown in Fig. 4.

From Fig. 4, we can observe that our proposed work excels, followed by the SMOTE-based methods (SBO SBAG and SENSG). The performance of the undersampling-based methods (RUS, UBAG, and EUS) is mediocre, and the worst performer is the hybrid-based method (EASY). The Friedman aligned-rank test is conducted, resulting in a significance level of p =0.000, which is low enough to reject the hypothesis of equivalence. Therefore, we continue with the Holm post-hoc test and present the adjust p-values in Table 5.

The proposed SNGEIP method was chosen as the control method, as it achieves the lowest average ranking in the Friedman test. The Holm test brings out the good performance of SNGEIP. SNGEIP statistically outperforms all the methods except for SBAG and SBO. As pointed by [46], sometimes the Friedman test reports a significant difference but the post-hoc test may fail to detect it, due to the lower power of the latter. For this reason, we use the Wilcoxon paired signed-rank test to compare the performance of SNGEIP SBAG and SBO. The test results have been reported in Table 6. The test rejects the hypothesis of equivalence with significance level at $p = 0.017$ and p =0.037, and hence, being the average ranking in favor of SNGEIP, its superiority over SBAG and SBO can be confirmed. Following all the results of the non-parametric statistical tests, we can state that SNGEIP outperforms the previous methods in the framework of imbalanced datasets. In addition to all these promising results, it's also worth pointing out that the method proposed in this paper provides significant better performance than SENSG. Both SNGEIP and SENSG have adopted the synthetic neighborhoods in their framework, but the synthetic samples in SENSG are composed of two parts: synthetic neighborhoods and synthetic samples generated by SMOTE. The superiority of our method over SENSG confirms that synthetic neighborhoods can bring more benefits than rebalancing the class distribution.

Finally, to visually show the advantage of SNGEIP with respect to the others, a scatter plot is provided in Fig. 5, where each point compares SNGEIP with one of other algorithms on a dataset. The x-coordinate of a point represents the AUC measure obtained by SNGEIP, whereas the y-coordinate represents the AUC measure obtained by other

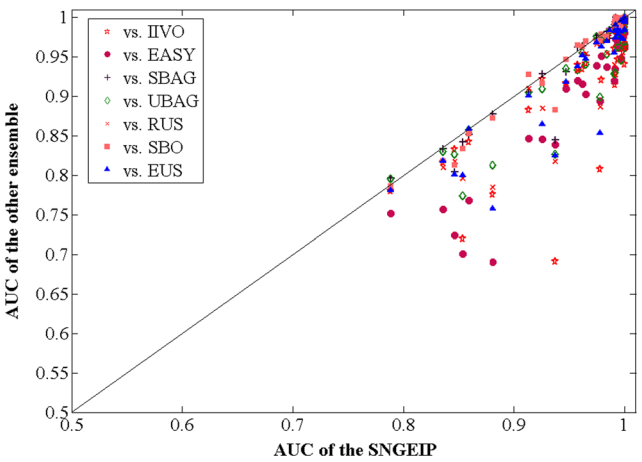

Fig. 5 Scatter plot of SNGEIP *vs*. other ensembles

Table 7 Average G_{mean} obtained from 3 × 5 cross validation for all the ensembles

Datasets	IR	Boosting-based			Bagging-based			Hybrid	
		EUS	SBO	RUS	UBAG	SBAG	SENSG	EASY	SNGEIP
ecoli0vs1	1.86	0.9727 ± 0.10	0.9784 ± 0.08	0.9693 ± 0.11	0.9737 ± 0.09	0.9591 ± 0.17	**0.9820 ± 0.02**	0.9630 ± 0.09	0.9819 ± 0.06
pimaimb	1.87	0.7279 ± 0.10	0.7345 ± 0.15	0.7187 ± 0.11	0.7516 ± 0.09	**0.7555 ± 0.11**	0.7417 ± 0.04	0.6960 ± 0.11	0.7501 ± 0.12
iris0	2.0	0.9825 ± 0.14	0.9825 ± 0.14	0.9825 ± 0.14	0.9825 ± 0.14	0.9720 ± 0.16	**0.9861 ± 0.03**	0.9825 ± 0.14	**0.9861 ± 0.12**
glass0	2.06	0.8238 ± 0.25	**0.8365 ± 0.29**	0.8284 ± 0.17	0.8153 ± 0.24	0.8266 ± 0.22	0.8058 ± 0.05	0.7622 ± 0.31	0.8116 ± 0.19
yeast1	2.46	0.7095 ± 0.11	0.7017 ± 0.14	0.6564 ± 0.15	0.7186 ± 0.08	**0.7208 ± 0.08**	0.7105 ± 0.03	0.6933 ± 0.12	0.7102 ± 0.12
vehicle2	2.88	0.9689 ± 0.05	**0.9834 ± 0.04**	0.9728 ± 0.04	0.9705 ± 0.05	0.9759 ± 0.03	0.9700 ± 0.01	0.9548 ± 0.05	0.9782 ± 0.04
vehicle3	2.99	0.7760 ± 0.08	0.7304 ± 0.16	0.7558 ± 0.17	0.7775 ± 0.13	0.7788 ± 0.15	0.7606 ± 0.03	0.7236 ± 0.19	**0.7789 ± 0.16**
glass0123vs456	3.20	0.9114 ± 0.18	0.9174 ± 0.19	0.9332 ± 0.18	0.9216 ± 0.21	0.9468 ± 0.14	0.9190 ± 0.05	0.9148 ± 0.19	**0.9494 ± 0.13**
vehicle0	3.25	0.9504 ± 0.05	**0.9720 ± 0.06**	0.9589 ± 0.06	0.9501 ± 0.05	0.9626 ± 0.05	0.9430 ± 0.03	0.9356 ± 0.06	0.9513 ± 0.06
ecoli1	3.36	0.8982 ± 0.15	0.8685 ± 0.25	0.8962 ± 0.20	0.8946 ± 0.12	0.8976 ± 0.14	0.8874 ± 0.03	0.8707 ± 0.15	**0.8993 ± 0.12**
newthyroid2	5.14	0.9432 ± 0.20	0.9734 ± 0.10	0.9584 ± 0.13	0.9459 ± 0.17	0.9462 ± 0.22	0.9428 ± 0.06	0.9486 ± 0.18	**0.9586 ± 0.16**
newthyroid1	5.14	0.9426 ± 0.18	**0.9769 ± 0.12**	0.9558 ± 0.12	0.9540 ± 0.13	0.9691 ± 0.14	0.9546 ± 0.05	0.9687 ± 0.11	0.9582 ± 0.13
ecoli2	5.46	0.8937 ± 0.15	0.9010 ± 0.19	0.9031 ± 0.17	0.9067 ± 0.18	0.9167 ± 0.19	0.8977 ± 0.05	0.8337 ± 0.24	**0.9186 ± 0.18**
segment0	6.02	0.9902 ± 0.03	**0.9933 ± 0.04**	0.9893 ± 0.03	0.9846 ± 0.03	0.9857 ± 0.04	0.9928 ± 0.01	0.9853 ± 0.03	0.9926 ± 0.03
glass6	6.38	0.9017 ± 0.18	0.9105 ± 0.18	0.8961 ± 0.31	0.8933 ± 0.24	0.9177 ± 0.19	0.9122 ± 0.06	0.8873 ± 0.19	**0.9197 ± 0.20**
yeast3	8.10	0.9227 ± 0.10	0.8701 ± 0.18	0.9074 ± 0.09	0.9323 ± 0.08	0.9274 ± 0.08	0.9229 ± 0.03	0.9010 ± 0.10	**0.9331 ± 0.08**
ecoli3	8.6	0.8572 ± 0.20	0.8067 ± 0.44	0.8168 ± 0.24	0.8715 ± 0.17	0.8408 ± 0.23	0.8122 ± 0.10	0.8580 ± 0.15	**0.8893 ± 0.21**
yeast2vs4	9.08	**0.9354 ± 0.11**	0.8857 ± 0.24	0.9141 ± 0.18	0.9178 ± 0.23	0.9269 ± 0.14	0.8819 ± 0.09	0.9112 ± 0.22	0.9095 ± 0.22
vowel0	9.98	0.9671 ± 0.07	0.9876 ± 0.06	0.9585 ± 0.09	0.9517 ± 0.13	0.9818 ± 0.12	0.9830 ± 0.03	0.9565 ± 0.11	**0.9914 ± 0.03**
glass016vs2	10.29	**0.7501 ± 0.54**	0.6852 ± 0.86	0.6412 ± 0.85	0.6733 ± 0.49	0.6306 ± 1.10	0.6429 ± 0.23	0.6499 ± 0.37	0.7040 ± 0.87
glass2	11.59	0.6753 ± 0.46	0.6856 ± 0.98	0.7158 ± 0.56	**0.7565 ± 0.37**	0.7309 ± 0.72	0.5817 ± 0.32	0.6575 ± 0.55	0.7471 ± 0.57
shutlec0vsc4	13.87	**1.0000 ± 0.00**	0.9997 ± 0.00	**1.0000 ± 0.00**	**1.0000 ± 0.00**	0.9996 ± 0.00	**1.0000 ± 0.00**	**1.0000 ± 0.00**	**1.0000 ± 0.00**
glass4	15.46	0.8132 ± 0.63	0.7884 ± 0.90	0.7817 ± 1.04	0.8603 ± 0.58	0.8949 ± 0.37	0.9102 ± 0.11	0.8529 ± 0.46	**0.9328 ± 0.38**
ecoli4	15.8	0.9125 ± 0.18	0.8987 ± 0.32	0.8967 ± 0.23	0.8737 ± 0.27	**0.9265 ± 0.25**	0.8200 ± 0.16	0.8795 ± 0.24	0.9083 ± 0.25
abalone9-18	16.40	0.6939 ± 0.33	0.6258 ± 0.46	0.5406 ± 0.79	**0.7395 ± 0.29**	0.6463 ± 0.38	0.6835 ± 0.14	0.6781 ± 0.41	0.6057 ± 0.45
glass016vs5	19.44	0.9582 ± 0.29	0.9633 ± 0.29	0.9265 ± 0.33	0.9265 ± 0.33	0.9701 ± 0.29	0.8683 ± 0.26	0.9035 ± 0.34	**0.9682 ± 0.28**
shuttlec2vsc4	20.5	0.9942 ± 0.08	**1.0000 ± 0.00**	0.9546 ± 0.40	0.9546 ± 0.40	0.9972 ± 0.03	**1.0000 ± 0.00**	0.9546 ± 0.40	**1.0000 ± 0.00**
glass5	22.78	**0.9733 ± 0.10**	0.8673 ± 1.01	0.9417 ± 0.13	0.9400 ± 0.12	0.8392 ± 1.31	0.9368 ± 0.12	0.9011 ± 0.30	0.9235 ± 0.96
yeast5	32.72	0.9475 ± 0.13	0.8916 ± 0.32	0.9421 ± 0.17	0.9462 ± 0.14	0.9619 ± 0.14	0.9406 ± 0.06	0.9420 ± 0.11	**0.9757 ± 0.09**
ecoli0137vs26	39.14	0.6979 ± 1.38	0.7006 ± 1.66	**0.7666 ± 0.93**	0.7254 ± 0.89	0.6719 ± 1.61	0.6198 ± 0.45	0.7391 ± 0.91	0.7123 ± 1.69
abalone17vs78910	39.31	0.7898 ± 0.24	0.6212 ± 0.40	0.6885 ± 0.41	**0.8234 ± 0.22**	0.7361 ± 0.29	0.7577 ± 0.08	0.7682 ± 0.30	0.6894 ± 0.45

Table 7 (continued)

Datasets	IR	Boosting-based			Bagging-based			Hybrid	
		EUS	SBO	RUS	UBAG	SBAG	SENSG	EASY	SNGEIP
krvskvs8	53.07	0.9721 ± 0.08	0.9840 ± 0.22	0.9487 ± 0.13	0.9498 ± 0.13	0.9673 ± 0.35	0.9793 ± 0.05	0.9531 ± 0.12	**1.0000 ± 0.00**
shuttle2vs5	66.67	0.9969 ± 0.05	**1.0000 ± 0.00**	0.9970 ± 0.04	0.9970 ± 0.04	**1.0000 ± 0.00**	**1.0000 ± 0.00**	0.9970 ± 0.04	0.9998 ± 0.00
kddcuplandvssatan	75.67	**1.0000 ± 0.00**	**1.0000 ± 0.00**	**1.0000 ± 0.00**	**1.0000 ± 0.00**	**1.0000 ± 0.00**	**1.0000 ± 0.00**	**1.0000 ± 0.00**	**1.0000 ± 0.00**
krvskvs15	80.22	0.9811 ± 0.06	**0.9998 ± 0.00**	0.9867 ± 0.05	0.9838 ± 0.06	0.9971 ± 0.01	**1.0000 ± 0.00**	0.9845 ± 0.05	0.9998 ± 0.00
kddcuprootkitimapvsback	100.13	0.9718 ± 0.31	**0.9998 ± 0.00**	0.9794 ± 0.30	0.9794 ± 0.30	**0.9998 ± 0.00**	**1.0000 ± 0.00**	0.9794 ± 0.30	0.9911 ± 0.13
Average		0.8945	0.8803	0.8800	0.8956	0.8931	0.8819	0.8774	**0.9010**

methods. Therefore, points that appear below the $y = x$ line correspond to the datasets where SNGEIP outperforms the other methods. From Fig. 5, we can observe that most of the points lie under the $y = x$ line. In addition, we find that when SNGEIP performs better, it usually provides much better AUCs than the benchmark algorithms (the corresponding points have higher distance to the $y = x$ line). On the contrary, when it provides worse performance, its loss is not so significant.

4.5.2 G-mean as a measure

Table 7 reports the average *G-means* obtained from the cross validation. The results are presents in the form of 'average ± standard deviation', and the bold-face indicates the best *G-mean* in each dataset. The results in Table 7 also confirm the superiority of our proposed method. Among the eight ensemble solutions for class imbalance problem, our method achieves highest number of *G-means* (on 17 of the 36 datasets), and highest average *Gmean* (0.9010). We conducted Friedman aligned-rank test on the obtained *G-means*, and presented average rankings in Fig. 6. From Fig. 6, we can find that the average rankings computed from the *G-means* are different from the ones computed from AUCs, but our proposed method still achieves the lowest average ranking. The Friedman test results in a significance level at $p = 0.000$, which rejects the hypothesis of equivalence. As there exists a significant difference, we continue with the Holm post-hoc test (Table 8). Still, our method has been chosen as the control method. Table 8 shows the average rankings of the algorithms and the adjusted p-values computed from the Holm test. As we can see, SNGEIP statistically outperforms all the other methods.

4.5.3 Efficiency

The average runtimes of all the ensemble solutions, except the SENSG method, on all the considered datasets are presented in Table 9. We didn't report the efficiency of SENSG because SENSG was tested on the Weka, a platform different from the other ensemble solutions. All the results were obtained on the same PC with an i5 CPU (2.7 GHz) and 8GB RAM. According to the average runtimes, all the algorithms can be classified into three groups. The first group consists of EASY, RUS and UBAG. These three algorithms combine the ensemble learning with undersampling techniques, and their base classifiers are trained on a shrinking of the original training data, so they present highest efficiency. The second group consists of SBO, SBAG and the proposed SNGEIP method. All these methods combine ensemble learning with oversampling techniques. Their base classifiers are trained on the union of the original training set and the synthetic data. Besides, it takes some extra time to

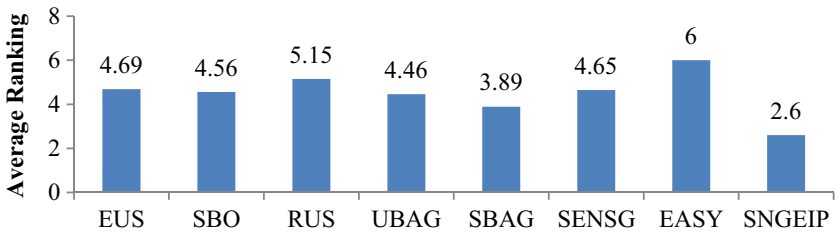

Fig. 6 Average rankings obtained from Friedman aligned-rank test based on the G_{mean}

generate the synthetic samples, so these three methods show lower efficiency. It is worth mentioning that our proposed method presents the lowest efficiency in the second group. It is because all the other two algorithms generate the minority samples only one time, on the contrary, our proposed method needs to generate the samples for each of the base classifiers iteratively. Among all the eight methods, EUS takes significant longer time than the other seven methods. This is because EUS adopts genetic algorithm to prepare the training set for each base classifier.

4.6 Parameters

We analyze whether the factors that account for the diversity help to improve the generalization ability of the final ensembles. In our proposed method, two parameters, the replacement ratio (*RR*) and λ, can determine the diversity of the final ensemble. The effects on diversity and AUCs of the varying parameter values are explored.

We follow a procedure that is similar to that outlined in [36], four datasets were randomly selected for analysis, based on variations in the number of samples, attributes, and imbalance ratio. We use the pairwise plain disagreement measure technique [49] to evaluate the diversity of our proposed method. The plain disagreement diversity for a base classifier pair C_i and C_j can be computed according to:

$$div_{i,j} == \frac{1}{N}\sum_{k=1}^{N} Diff(C_i(x_k), C_j(x_k)) \tag{15}$$

Table 8 Holm post-hoc test results based on G_{mean}

Control method (SNGEIP)

	Algorithm (ranking)	Z	p-value	Holm	Hypothesis (α =0.05)
7	EASY (6)	5.8938	0.0000	0.0071	**Rejected**
6	RUS (5.15)	4.4264	0.0000	0.0083	**Rejected**
5	EUS(4.69)	3.6325	0.0005	0.0100	**Rejected**
4	SENSG (4.65)	3.5603	0.0007	0.0125	**Rejected**
3	SBO (4.56)	3.3919	0.0013	0.0167	**Rejected**
2	UBAG(4.46)	3.2235	0.0022	0.0250	**Rejected**
1	SBAG(3.89)	2.2372	0.0327	0.0500	**Rejected**

The bold emphasis indicates the significant statistical test results

Where N is the number of training samples, and $C_i(x_k)$ is the classification result assigned by the classifier C_i to sample $\boldsymbol{x_k}$. Here $Diff(a, b) = 0$ if $a = b$, otherwise $Diff(a, b) = 1$. The average value of diversity between all pairs of classifiers is used as the diversity measure for the ensemble.

A 5-fold CV was adopted to divide the dataset into training parts and testing parts. The training parts were used to learn the classifiers and calculate the diversity. Testing parts were used to calculate the AUCs. The diversity measures and AUCs obtained from the whole procedure are then averaged to produce a single measure. Figure 7 shows the obtained diversities and AUCs under different setting of *RR* and λ. In order to study the parameters' influence separately, one parameter was fixed when studying the other one. Specifically, when studying the influence of *RR*, we set $\lambda = 0.5$ and varied the value of *RR* from 0.1 to 0.9. The first and second rows of Fig. 7 show the effect of *RR* on the classification ability and diversity. On the other hand, when studying the effect of λ, we set $RR = 0.3$, and varied the value of λ from 0.1 to 0.9. The third and fourth rows of Fig. 7 show the effect of λ on classification ability and diversity.

Considering all the experimental results in Fig. 7, the following conclusions and analyses can be provided.

(1) Greater diversity can be achieved by assigning a larger value to parameter *RR* or λ. Larger values of *RR* and λ indicate the generated samples can be more different from the original training samples, and the base classifiers will be more diverse. The second and forth rows of Fig. 7 show the influence of parameters on the

Table 9 Average runtime (seconds) for all the ensembles

Rank.	Algorithm	Average runtime (s)
1	EASY	7.31
2	RUS	8.75
3	UBAG	25.58
4	SBO	52.63
5	SBAG	142.23
6	SNGEIP	395.26
7	EUS	2043.5

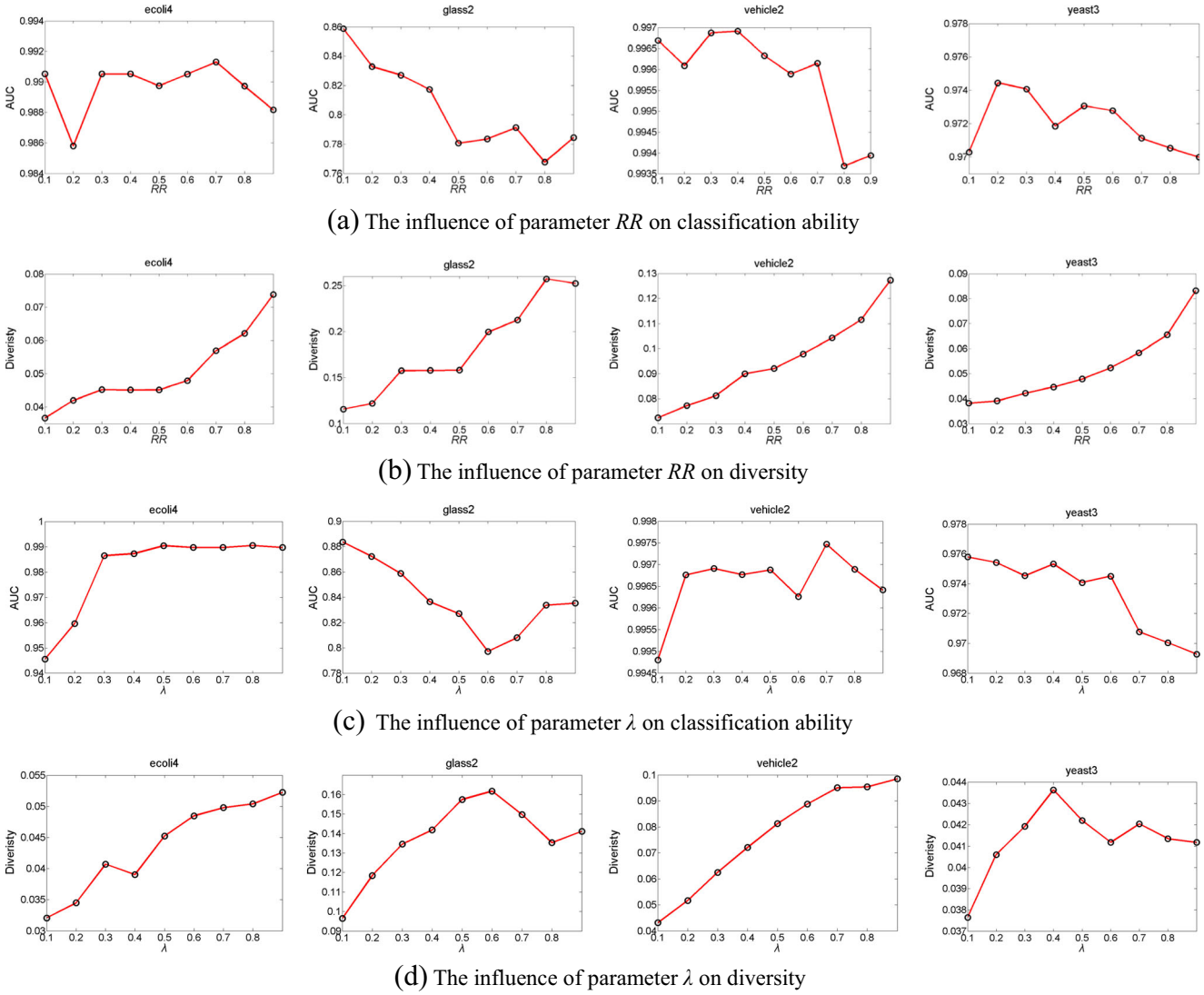

Fig. 7 Influence of replacement ratio (*RR*) and λ on classification ability and diversity

diversity. In most of the cases, the diversity increases monotonically by increasing the parameter *RR* and λ. This indicates a desirable level of diversity can be achieved by setting proper values for parameter *RR* and λ.

(2) Increasing the diversity is not always beneficial. A fundamental issue toward designing an optimal ensemble learning is how to address the conflict between increasing diversity and decreasing individual error. Sample-generation based methods try to address this conflict by generating very different samples and training the base classifier with the generated samples along with the original ones. Although greater diversity can be achieved by these methods, AKHAND et al. [38] have pointed out that greater diversity achieved by these methods does not necessarily lead to a better classification ability. The first and third rows of Fig. 7 confirm that. In first and third rows of Fig. 7, the AUC does not improve along with the increase in the diversity. Once parameter *RR* or λ reaches a certain level, the AUC does not improve. This seems to violate the motivation for promoting diversity. However, the experimental results in last section confirms that significant better performance can be achieved by setting proper parameters, for example $RR = 0.368$ and $\lambda = 0.5$.

The setting of parameter *RR* and λ is the trade-off between increasing diversity and decreasing individual error. It requires appropriately setting for these two parameters. For this, parameter setting in our experiments is given. All the results in Tables 4 and 7 were achieved under the setting of $RR = 0.368$ and $\lambda = 0.5$. The proposed method is variant of Bagging method [27]. In Bagging [27], the

Table 10 Variations of AUC and diversity during the grid search procedure

Dataset	AUC		Diversity	
	RR	λ	*RR*	λ
Ecoli4	0.0055	**0.0450**	**0.0371**	0.0202
Glass2	**0.0913**	0.0865	**0.1417**	0.0653
Vehicle2	**0.0032**	0.0027	0.0550	**0.0553**
Yeast3	0.0045	**0.0065**	**0.0449**	0.0060
Average	0.0261	0.0352	0.0697	0.0367

The bold emphasis indicates more dramatic change in the diversity or AUC

training set for each base classifier is randomly sampled from the original training set with replacement, and a particular training data has a probability of $(1\text{-}1/n)^n \approx 0.368$ of not being picked. In Bagging, each base classifier is trained with about 63.2% of the original training data. We set $RR = 0.368$, so that each base classifier in our proposed method can be trained with 63.2% of the original training data and 36.8% of augmented data. With such a setting, we can expect a significant improvement in performance compared to Bagging. As for the parameter λ, it is a trade-off between increasing diversity and decreasing individual error of base classifier. By now, there is no theoretic way of computing the λ value. We can select the optimal λ by using grid search method, or set it by experience. In this paper, we set $\lambda = 0.5$. This setting was determined by trial-and-error method.

However, based on our observation, better generalization ability can be ahieved by conducting a parameter optimization. An optimal *RR* can be chosen from the range [0.1, 0.9], and an optimal λ can be chosen from the range [0.3, 1]. These two ranges are much narrower than those of other classifiers, such as SVMs. The parameters can be set as the ones suggested in our experiments, or be selected by a standard grid search procedure, with the AUC obtained from *k*-fold cross validation as its generalization estimation. It is also worth mentioning that the parameter *RR* deserves more attention during a grid search procedure. Table 10 presents the variations of AUC and diversity during a standard grid search procedure. The results in Table 10 suggest that the change of parameter *RR* can lead to more dramatic change of the diversity within the final ensemble.

5 Discussion and conclusion

Recent developments in science and technology have enabled the growth and availability of raw data to occur at an explosive rate. While this data provides the opportunity of extracting knowledge that is impossible to get before, the increased prevalence of class imbalance problems in various real-world domains also presents a great challenge to extract knowledge for future prediction. Ensemble learning embedded with various data sampling techniques have proven to be a powerful way of addressing this challenge. However, in order to construct an optimal ensemble learning for the class imbalance problems, the conflict between increasing diversity and decreasing individual error needs to be properly addressed.

In this paper, we introduce a new ensemble learning for the class imbalance problem that addresses the class imbalance problem and maintain proper diversity simultaneously. Inspired by the L-GEM [18], a special type of samples, called the neighborhoods of the training samples, is synthetized and imported into the ensemble learning. Experimental results on 36 public imbalanced datasets prove that our proposed method can produce promising results in this unfavorable scenario and significantly outperform previous state-of-the-art ensemble learning.

Although the strength of our method has been proven, there also exist some limitations in the current research. First, we only test the idea of synthetic neighborhood generation on the ensemble of decision tress. Ensemble of other base classifiers, such as SVMs and neural networks, should also be tested. Second, we do not use any sophisticated ensemble pruning technique to remove the useless base classifiers in the final ensemble Third, the classification ability of our method largely relies on the quality of synthetic neighborhoods, so other method should be studied to generate more qualified synthetic neighborhoods.

Acknowledgements The research is partly supported by Science and Technology Supporting Program, Sichuan Province, China (2013GZX0138 and 2014GZ0154).

References

1. López V, Fernández A, García S, Palade V, Herrera F (2013) An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics. Inf Sci 250:113–141
2. Galar M, Fernandez A, Barrenechea E, Bustince H, Herrera F (2012) A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans Syst Man Cybern Part C Appl Rev 42(4):463–484
3. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
4. Yang Q, Wu X (2006) 10 Challenging problems in data mining research. Int J Inf Technol Decis Mak 05(04):597–604
5. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Int Res 16(1):321–357
6. Han H, Wang W-Y, Mao B-H (2005) Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Advances in intelligent computing. Springer, pp 878–887

7. Barua S, Islam MM, Yao X, Murase K (2014) MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning. IEEE Trans Knowl Data Eng 26(2):405–425
8. Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C (2011) DBSMOTE: density-based synthetic minority over-sampling technique. Appl Intell 36(3):664–684
9. Liu X-Y, Wu J, Zhou Z-H (2009) Exploratory undersampling for class-imbalance learning. Trans Syst Man Cybern Part B 39(2):539–550
10. Galar M, Fernández A, Barrenechea E, Herrera F (2013) EUSBoost: enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. Pattern Recognit 46(12):3460–3471
11. Yu H, Ni J, Zhao J (2013) ACOSampling: an ant colony optimization-based undersampling method for classifying imbalanced DNA microarray data. Neurocomput 101:309–318
12. Qian Y, Liang Y, Li M, Feng G, Shi X (2014) A resampling ensemble algorithm for classification of imbalance problems. Neurocomputing 143:57–67
13. Stefanowski J, Wilk S (2008) Selective pre-processing of imbalanced data for improving classification performance. In: Song I-Y, Eder J, Nguyen TM (eds) Data warehousing and knowledge discovery: 10th international conference, DaWaK 2008 Turin, Italy, September 2–5, 2008 Proceedings. Springer, Berlin, pp 283–292
14. Sun Z, Song Q, Zhu X, Sun H, Xu B, Zhou Y (2015) A novel ensemble method for classifying imbalanced data. Pattern Recognit 48(5):1623–1637
15. Kittler J, Hatef M, Duin RPW, Matas J (1998) On combining classifiers. IEEE Trans Pattern Anal Mach Intell 20(3):226–239
16. Díez-Pastor JF, Rodríguez JJ, García-Osorio CI, Kuncheva LI (2015) Diversity techniques improve the performance of the best imbalance learning ensembles. Inf Sci 325:98–117
17. Visentini I, Snidaro L, Foresti GL (2016) Diversity-aware classifier ensemble selection via f-score. Inf Fusion 28:24–43
18. Yeung DS, Ng WW, Wang D, Tsang EC, Wang X-Z (2007) Localized generalization error model and its application to architecture selection for radial basis function neural network. IEEE Trans Neural Netw 18(5):1294–1305
19. Ng WWY, Dorado A, Yeung DS, Pedrycz W, Izquierdo E (2007) Image classification with the use of radial basis function neural networks and the minimization of the localized generalization error. Pattern Recognit 40(1):19–32
20. Ng WWY, Yeung DS, Firth M, Tsang ECC, Wang X-Z (2008) Feature selection using localized generalization error for supervised classification problems using RBFNN. Pattern Recognit 41(12):3706–3719
21. Chen Z, Lin T, Chen R, Xie Y, Xu H (2017) Creating diversity in ensembles using synthetic neighborhoods of training samples. Appl Intell 47(2):570–583
22. Weiss GM, Tian Y (2008) Maximizing classifier utility when there are data acquisition and modeling costs. Data Min Knowl Disc 17(2):253–282
23. Sun Y, Kamel MS, Wong AKC, Wang Y (2007) Cost-sensitive boosting for classification of imbalanced data. Pattern Recognit 40(12):3358–3378
24. Wang S, Yao X, IEEE (2009) Diversity analysis on imbalanced data sets by using ensemble models. In: 2009 IEEE symposium on computational intelligence and data mining. IEEE, New York, pp 324–331
25. Seiffert C, Khoshgoftaar TM, Van Hulse J, Napolitano A (2010) RUSBoost: a hybrid approach to alleviating class imbalance. IEEE Trans Syst Man Cybern Part A Syst Hum 40(1):185–197
26. Barandela R, Sanchez JS, Valdovinos RM (2003) New applications of ensembles of classifiers. Pattern Anal Appl 6(3):245–256
27. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
28. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139
29. Chawla NV, Lazarevic A, Hall LO, Bowyer KW et al (2003) SMOTEBoost: improving prediction of the minority class in boosting. In: Lavrač N (ed) Knowledge discovery in databases: PKDD 2003: 7th European conference on principles and practice of knowledge discovery in databases, Cavtat-Dubrovnik, Croatia, September 22–26, 2003. Proceedings. Springer, Berlin, pp 107–119
30. Freund Y (1996) Experiments with a new boosting algorithm. In: Thirteenth international conference on machine learning
31. Bhowan U, Johnston M, Zhang M, Yao X (2014) Reusing genetic programming for ensemble selection in classification of unbalanced data. IEEE Trans Evol Comput 18(6):893–908
32. Díez-Pastor JF, Rodríguez JJ, García-Osorio C, Kuncheva LI (2015) Random balance: ensembles of variable priors classifiers for imbalanced data. Knowl-Based Syst 85:96–111
33. Melville P, Mooney RJ (2005) Creating diversity in ensembles using artificial data. Inf Fusion 6(1):99–111
34. Martínez-Muñoz G, Suárez A (2005) Switching class labels to generate classification ensembles. Pattern Recognit 38(10):1483–1494
35. Ho TK (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):832–844
36. Akhand MA, Murase K (2012) Ensembles of neural networks based on the alteration of input feature values. Int J Neural Syst 22(1):77–87
37. Brown G, Wyatt J, Harris R, Yao X (2005) Diversity creation methods: a survey and categorisation. Inf Fusion 6(1):5–20
38. Akhand MAH, Islam MM, Murase K (2009) A comparative study of data sampling techniques for constructing neural network ensembles. Int J Neural Syst 19(02):67–89
39. Sun B, Ng WWY, Yeung DS, Chan PPK (2013) Hyper-parameter selection for sparse LS-SVM via minimization of its localized generalization error. Int J Wavelets Multiresolution Inf Process 11(03):1350030
40. Zhang H, Li M (2014) RWO-sampling: a random walk oversampling approach to imbalanced data classification. Inf Fusion 20:99–116
41. Alcalá-Fdez J, Sánchez L, García S, del Jesus MJ, Ventura S, Garrell JM, Otero J, Romero C, Bacardit J, Rivas VM, Fernández JC, Herrera F (2009) KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput 13(3):307–318
42. Chang CC, Lin CJ (2007) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2(3, article 27):389–396
43. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. SIGKDD Explor Newsl 11(1):10–18
44. Bradley AP (1997) The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognit 30(7):1145–1159
45. Huang J, Ling CX (2005) Using AUC and accuracy in evaluating learning algorithms. IEEE Trans Knowl Data Eng 17(3):299–310
46. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7(1):1–30
47. Hodges JL, Lehmann EL (1962) Rank methods for combination of independent experiments in analysis of variance. Ann Math Stat 33(2):482–497
48. Holm S (1979) A simple sequentially rejective multiple test procedure. Scand J Stat 6(2):65–70
49. Tsymbal A, Pechenizkiy M, Cunningham P (2005) Diversity in search strategies for ensemble feature selection. Inf Fusion 6(1):83–98