BPSO-Adaboost-KNN ensemble learning algorithm for multi-class imbalanced data classification

Article *in* Engineering Applications of Artificial Intelligence · October 2015

DOI: 10.1016/j.engappai.2015.09.011

Engineering Applications of Artificial Intelligence 49 (2016) 176–193

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

BPSO-Adaboost-KNN ensemble learning algorithm for multi-class imbalanced data classification

Guo Haixiang [a,b,c,d,*], Li Yijing [a,b], Li Yanan [a,b], Liu Xiao [a,b], Li Jinling [a]

[a] *College of Economics and Management, China University of Geosciences, Wuhan 430074, China*
[b] *Research Center for Digital Business Management, China University of Geosciences, Wuhan 430074, China*
[c] *Mineral Resource Strategy and Policy Research Center of China University of Geosciences, Wuhan 430074, China*
[d] *School of Business, Central South University, Changsha, Hunan 410083, China*

ARTICLE INFO

Article history:
Received 8 April 2015
Received in revised form
14 August 2015
Accepted 28 September 2015
Available online 29 October 2015

Keywords:
Imbalanced data
Ensemble
Feature selection
Classification
Oil reservoir

ABSTRACT

This paper proposes an ensemble algorithm named of BPSO-Adaboost-KNN to cope with multi-class imbalanced data classification. The main idea of this algorithm is to integrate feature selection and boosting into ensemble. What's more, we utilize a novel evaluation metric called AUCarea which is especially for multi-class classification. In our model BPSO is employed as the feature selection algorithm in which AUCarea is chosen as the fitness. For classification, we generate a boosting classifier in which KNN is selected as the basic classifier. In order to verify the effectiveness of our method, 19 benchmarks are used in our experiments. The results show that the proposed algorithm improves both the stability and the accuracy of boosting after carrying out feature selection, and the performance of our algorithm is comparable with other state-of-the-art algorithms. In statistical analyses, we apply Bland–Altman analysis to show the consistencies between AUCarea and other popular metrics like average G-mean, average *F*-value etc. Besides, we use linear regression to find deeper correlation between AUCarea and other metrics in order to show why AUCarea works well in this issue. We also put out a series of statistical studies in order to analyze if there exist significant improvements after feature selection and boosting are employed. At last, the proposed algorithm is applied in oil-bearing of reservoir recognition. The classification precision is up to 99% in oilsk81-oilsk85 well logging data in Jianghan oilfield of China, which is 20% higher than KNN classifier. Particularly, the proposed algorithm has significant superiority when distinguishing the oil layer from other layers.

1. Introduction

Classification is one of the crucial issues in the field of machine learning. Though many mature classification methods have been proposed since the last century such as Decision Tree, Naïve Bayes, Artificial Neural Network (ANN), K-Nearest Neighbor (KNN) and Support Vector Machine (SVM), most of those approaches are based on the assumption that the sample distribution among various classes are balanced. When facing imbalanced distribution, the traditional classifiers often come up to a disappointed performance (Galar et al., 2013). Therefore, traditional classifiers remain to be modified in addressing the problem of imbalanced data classification.

Imbalanced data refers to such a dataset in which one or some of the classes have much more samples in comparison to the others. The class that has far more samples is called the majority class, while a class that contains relative small number of samples is called minority class (earle, 1987). When addressing imbalanced data problems, people tend to care more about the minority class, and the cost of misclassifying those samples belong to minority class are much higher than the others (Menardi and Torelli, 2014; López et al., 2013; Maldonado et al., 2014). Taking cancer diagnosis for example, the number of cancer patient is much less than healthy people, if cancer patients are diagnosed as healthy people, they will miss the best treatment time, which may cause a disaster (Menardi and Torelli, 2014). So does oil-bearing recognition (oil-bearing recognition means to recognize the characters of each layer in the well (Feng et al., 1999; Guo et al., 2011)) that is studied in this paper, the class distribution of logging data is imbalanced and cost of misclassifying oil layer into other layers is much higher than other misclassification situations. Therefore, oil-bearing recognition is a typical imbalanced data classification problem.

The approaches to deal with imbalanced data recognition can be summarized into the following three categories:

* Corresponding author at: College of Economics and Management, China University of Geosciences, Wuhan 430074, China. Tel.: +86 15927389298; fax: +86 027 67883201.
E-mail address: faterdumk0732@sina.com (G. Haixiang).

http://dx.doi.org/10.1016/j.engappai.2015.09.011
0952-1976/© 2015 Elsevier Ltd. All rights reserved.

- Data level approaches: data level approaches focus on resizing the training datasets in order to balance all kinds of classes. Two main ideas of resizing are over-sampling and under-sampling, which are all based on sampling technologies. Dozens of over-sampling and under-sampling algorithms have been proposed before, such as Chawla put forward an algorithm called SMOTE (Chawla et al., 2002) to over-sample the instances of minority class and Dehmeshki proposed the data filtering technology (Vorraboot et al., 2015) to under-sample the samples of majority class.
- Algorithmic level approaches: algorithmic level approaches focus on carring out modification on existing algorithms/classifiers to strengthen their ability of learning from minority class (Galar et al., 2012; López et al., 2013). In this family there are various commonly used methods such as cost sensitive and one-class learning. (Galar et al., 2012; Cao et al., 2013). Cost sensitive algorithms attempt to increase the learning ability of classifiers by assigning larger misclassified cost for minority class samples. López et.al. (2015) proposed Chi-FRBCS-BigDataCS algorithm to deal with large-scale imbalanced data using a fuzzy rule and cost-sensitive learning techniques. Krawczyk et.al. (2014) constructed a fusion algorithm based on cost-sensitive decision tree ensembles, in which choice of cost matrix is estimated by ROC analysis. Nguyen et.al introduced two empirical cost-sensitive algorithms, one combined sampling, cost-sensitive and SVM and the other treated the cost ratio as a hyperparameter which needs to be optimized before training the final model (Thai-Nghe et al., 2010). One-class classifier only models single class by defining a boundary to classify binary classes datasets. According to different approaches of computing the boundary, one-class learning can be divided into the method based on density, neural network, clustering and SVM (Maldonado et al., 2014; Chawla et al., 2004; Naimul et al., 2014). However, the most popular one-class classifier that is used for imbalanced data classification is One-Class SVM(OSVM). Krawczyk et al. (2014) proposed a weighted OSVM to model the minority class. Tian et al. (2011) employed a SMOTE-OSVM algorithm to detect noises of minority samples, utilized the oversampled minority samples and a partial of majority samples to train SVM classifier. Krawczyk and Woźniak (2015) introduced an ensemble classifier, in which OSVM is used.
- New evaluation metrics: traditional evaluation metrics such as classification accuracy may ignore the minority class or treat them as noise (Galar et al., 2013; Menardi and Torelli, 2014). Performance metrics adapted into imbalanced data problems, such as Receiver Operating Characteristics (ROC) (Richard and Jonathan, 2006), G-mean, and *F*-value (López et al., 2013), are less likely to suffer from imbalanced distributions because they have less bias toward majority class. In fact most of algorithms cope with class imbalanced problems have abandoned using accuracy as performance metric, while ROC, G-mean and *F*-value are most popular used (López et al., 2013, 2015; Sun et al., 2015; Krawczyk et al., 2014). Recently, many modified metrics are proposed to evaluate the performance of imbalanced data classification problems. Gao et al. (2014) built several neuro-fuzzy models for imbalanced data classification, aimed to minimize the leave-one-out(LOO) mean square error(MSE). All of the neurofuzzy models used modified AUC and *F*-value as performance metrics called LOO-AUC and LOOFM, since the basic AUC and *F*-value are no longer applicable if MSE is chosen as the optimization goal. Krawczyk and Schaefer (2013) proposed a multiple classifier system, in which double-fault diversity measure is considered to prune those base classifiers that output similar hypotheses. For multi-class imbalanced problems, basic imbalanced data metrics also need to be modified due to most of those metrics are designed for addressing binary-class problems.

However, in recent years, several ensemble algorithms combined with different stratergies are proposed to cope with imbalanced data classification. Most of the ensemble algorithms are based on two specific algorithms, which are, the Boosting algorithm proposed by Schapire (Freund, 1995) and Bagging proposed by Breiman (1996). Sun et.al., (2015) proposed a novel ensemble strategy for imbalanced data classification, which converts an imbalanced dataset into multiple balanced subset and gets final hypothesis using an ensemble classifier , similar strategy can be found in (Díez-Pastor et al., in press). Nitesh et al. (2003) puts forward an algorithm called SMOTEBoost, which integrate over-sampling method SMOTE and boosting algorithm. Krawczyk and Schaefer, (2013) created an ensemble algorithm called PUSBE that contains sampling, pruning and boosting technologies. Nanni et al. (2015) denoted that these approaches generally contain 3 processes, that are, resampling, ensemble building, and voting for the final classification. In this family, ensemble algorithms like EasyEnsemble (Liu et al., 2009), EUSboost (Galar et al., 2013) are all well-accepted state-of-the-art. In this paper we also utilize boosting algorithm as classifier due to its efficiency in dealing with imbalanced problems.

Instead of proposing new methods, Prati et al. (2015) focused on setting up a series of experiments to assess the performance of some proposed treatment methods like SMOTE (Chawla et al., 2002), ADASYN (He et al., 2008) and MetaCost (Wang et al., 2013) for imbalanced data. These treatment methods are all based on sampling technologies and cost sensitive learning, which are considered as the powerful methods to increase the learning ability of classifier to classify the minority class. In their research, they defined a value called “performance loss” to figure out whether all the learning models are equally affected by class imbalanced. Besides, they also defined a metric called “performance recovery” to evaluate how much of the performance losses caused by imbalanced distribution can be recovered by the treatment methods. The results showed not all the treatment methods are suit for all basic algorithms. For example, SMOTE are considered as the most common sampling method for imbalance data, but it seems to harm the performance of SVM and Naïve Bayes. This idea inspired us to find new treatments for imbalanced data instead of sticking on sampling methods.

While most of literatures focused on binary problems, Alberto et al. (2013) reviewed plenty of novel multi-class imbalanced data classification algorithms. They pointed out that multiple class classification might achieve a lower performance than binary classification as the boundaries among the classes may overlap. They studied two ways of extending binary classification algorithms into multi-class case: One-versus-one approach (OVO) and One-versus-all approach (OVA). It shows in their experimental studies that OVO approaches gain better results than OVA approaches. However, when comparing pairwise learning with standard ad-hoc learning algorithms (algorithms that are natural for addressing multiple class learning problems), there is no significant evidence to illustrate that pairwise learning algorithms are superior to standard ad-hoc learning algorithms. Considering the computational cost of pairwise learning is expensive, we would like to utilize standard ad-hoc approaches instead of pairwise learning.

Feature selection is often separated as another topic for imbalanced data problems, as is discussed in several literatures like (Maldonado et al., 2014; Yin et al., 2013; Shanab et al., 2011). These literatures all focus on developing novel feature selection algorithms, while the contribution of feature selection for imbalanced data classification is not clearly discussed. Krawczyk and Schaefer (2013) employed FCBF feature selection algorithm as a pre-processing procedure before carrying out the classification model, which gained good experimental results.

Unlike most of ensemble methods that integrated sampling technologies or cost-sensitive methods into ensemble, our ensemble model has three typical contributions, as the following:

- First of all, we employ feature selection parallel with classification, not as a preprocessing as is suggested in Krawczyk and Schaefer (2013). It is obvious that removing irrelevant and redundant features can reduce the noise in the training space and decrease the time complexity (Yin et al., 2013; Alibeigi et al., 2012). For imbalanced case, samples in minority class are more easily to be treated as noise, if we remove the irrelevant features in the feature space, the risk of treating minority samples as noise may also be reduced. Since it is hard to judge the correlation between features and targets, we prefer to use wrapper method instead of filter method for feature selection.
- We choose to implement the powerful boosting framework Adaboost as the learning model, in which KNN is chosen as the base classifier and boosting-by-resample method is used to generate the training set.
- We aim to cope with multi-class imbalanced problems, while most of previous methods are designed for binary-class problems. Since most basic performance metrics such as AUC are binary metrics, we applied a new multi-class AUC metric called AUCarea to our model by setting probabilistic outputs for both KNN and Adaboost.

Therefore, in this paper, we develop an ensemble algorithm called BPSO-Adaboost-KNN that makes a fusion of feature selection, boosting and novel evaluation metric. BPSO-Adaboost-KNN firstly implements BPSO as the feature selection algorithm and then designs an Adaboost-KNN classifier to convert the traditional weak classifier KNN into a strong classifier. Finally, Both of these two processes in BPSO-Adaboost-KNN are measured by a new evaluation metric called AUCarea in order to pick best feature subset as well as final hypothesis of the ensemble model.

The outline of this paper is as follows: Adaboost-KNN classifier, BPSO, AUCarea and our ensemble algorithm BPSO-Adaboost-KNN are described in Section 2. Section 3 presents experimental results and comparisons between different classifiers and ensemble strategies using UCI and KEEL datasets, where statistical analyses are carried out to clarify the superiority of AUCarea used in our model. In Section 4 we apply our ensemble model to oil-bearing of reservoir recognition. At last, conclusions are placed in Section 5.

2. BPSO-Adaboost-KNN ensemble learning algorithm

In this section, Adaboost-KNN classifier is described at Section 2.1, in which we also explain the reasons that we choose KNN as the basic classifier. The new metric AUCarea is showed in Section 2.2. After that, we give a brief introduction of feature selection algorithm BPSO in Section 2.3. At last, the entire model is displayed in Section 2.4, in which the superiority of our model is analyzed.

2.1. *Adaboost-KNN classification algorithm*

2.1.1. *Adaboost series algorithms*

Boosting is an effective tool to improve the learning ability of basic learning algorithms. It is regarded as the most common and effective method in ensemble learning. The first applicable approach of boosting is Adaboost proposed by Freund and Schapire (1996), which has been rated as one of the top ten algorithms in data mining (Cao et al., 2013).

The basic Adaboost classification algorithm is implemented for binary classification problems, the main idea of Adaboost is as follows: A distribution D on training space is generated in which each training sample is assigned a weight of $1/m$ initially, where m is the number of training samples. Adaboost works by sequentially applying a weak classifier to train the reweighed training dataset (generated by the distribution D) and taking a majority vote to integrate all the weak hypotheses generated by weak classifiers into the final hypothesis. In each iteration, the sample distribution D is updated according to the hypothesis generated in this iteration. The rule of updating is, samples that failed to be assigned to the right class gain higher weights, so that in the next iteration the classifier will focus more on learning those failed classified samples.

To solve different kinds of classification problems, AdaBoost.M1, AdaBoost.M2 (Freund and Schapire, 1997), AdaBoost.MR, AdaBoost.MH (Schapire and Singer, 1999) have been proposed as the extensions of basic Adaboost algorithm, in which AdaBoost.M1, AdaBoost.M2 are used to solve multi-class with single-label problems, the later two are used to solve multi-class with multi-label problems. AdaBoost.M1 is the first extension to multi-class classification with a different weight changing mechanism. Adaboost.M1 algorithm is easy to implement, as long as the performance of each weak hypothesis is slightly better than random guessing (Freund and Schapire, 1996). Since in this paper we are looking forward to dealing with multi-class data with single-label, so that Adaboost.M1 algorithm is chosen in our model. The main processes of AdaBoost.M1 are outlined in Algorithm 1.

Algorithm 1. AdaBoost.M1 classification algorithm

```
Input: Training set $S=\{x_i, y_i\}$, $i=1\ldots m$, $y_i \in Y$, $Y=\{c_1, c_2, \ldots c_k\}$,
  $c_k$ is the class label; The number of iterations $T$; Weak classifier $I$.
Initializing weights distribution to $D_1(i)=1/m$
For $t=1$ to $T$
    Train classifier $I(S, D_t)$, get weak hypothesis
  $h_t = X \rightarrow \{c_1, c_2, \ldots c_k\}$
    Compute the error rate of $h_t$, $\varepsilon_t \leftarrow \sum_{i=1}^{m} D_t(i)[y_i \neq h_t(x_i)]$
    If $\varepsilon_t > 0.5$ then
        $T \leftarrow t-1$
        Continue
    End if
    Set $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$
  For $i=1$ to $m$
        update weight $D_{t+1}(i) = D_t(i){\beta_t}^{1-[y_i \neq h_t(x_i)]}$
  End for $i$
End for $t$
Output: the final hypothesis
  $H(x) = \arg\max\left(\sum_{t=1}^{T} \ln\left(\frac{1}{\beta_t}\right)[Y = h_t(X)]\right)$
```

2.1.2. *Basic classifier-KNN*

KNN (K-Nearest Neighbor) is a very simple but effective classifier. In this paper, we choose KNN as the weak classifier in Adaboost. The reasons that we decided to use KNN as basic classifier lie in the following two points:

- Computational cost: As we can see, boosting is an iterative algorithm, which means the computational complexity of Adaboost is T times of the computational complexity of basic classifier. Besides, we aim to carry out feature selection procedure parallels with boosting, which leads to an even higher time complexity. Therefore, the most crucial criteria for choosing basic classifier is to choose the simplest algorithm. KNN is our first choice since the computational cost of KNN is O(dN), where N is the number of training samples and d is the number of dimentions. In general, the most of implements of Adaboost prefer to use C4.5 or SVM as basic classifier (Galar et al., 2013; Sun et al., 2015; Nitesh et al., 2003; Krawczyk and Schaefer, 2013; Gao et al., 2014), for the reason that they are more powerful and can be generalized to use the given distribution directly. However, the computational costs of C4.5 and SVM are far more expensive than KNN. The run-time complexity of C4.5 is $O(N * d * \log d)$ and the run-time complexity of standard SVM for training and testing are $o(N_{sv}^3 + N * N_{SV}^2)$ and $O(N_{SV})$ respectively, where N_{sv} denotes the number of surport vectors (previous research shows N_{sv} grows as a linear function of N, see (Sathiya et al., 2006)).

- Natural for multi-class case: Since we focus on multi-class classification problem, another criteria for choosing basic classifier is to find a classifier that is natural in dealing with multi-class case. Although most of binary classifiers can be generalized to multi-class, they all add complexity of the model (like use SVM based on OVO or OVA approach). Considering that, KNN is natural and simple enough that make it become a good choice.

Boosting is deigned for use with any learning algorithms, the main concern is that the training samples now have varying weights (Freund and Schapire, 1996). There are two main approaches to address these weights: boosting by resampling and boosting by reweighting (Freund and Schapire, 1997). As KNN cannot be generalized to use a given distribution directly, we choose to use boosting by resampling in our model. The resample algorithm is called FiltEX, which is described in Algorithm 2, the main idea can be found in (Freund, 1995).

Algorithm 2. FiltEX Algorithm

```
1. Choose a real number $x$ uniformly at random in the range $0 \le x < 1$
2. Perform a binary search for the index $j$ for which $\sum_{i=1}^{j-1} w_i \le x < \sum_{i=1}^{j} w_i$
3. Return the example $(x_i, y_i)$
```

2.2. Evaluation metric

Accuracy is the most commonly used evaluation metric for classification. However, for imbalanced data classification problems, accuracy may not be a good choice because accuracy often has a bias toward majority class (Martino et al., 2013; López et al., 2006). Taking cancer diagnosis for example. Assuming that our training data consist 99% of healthy people and 1% patients. If all the samples are classified as healthy people by a classifier, then the accuracy is up to 99%, which seems not bad. However, in fact, there is no practical meaning exists if the classifier cannot identify the patients. Thereby, some other metrics need to be considered for imbalanced data classification problems. ROC (Receiver Operating Characteristic) curve is recognized as the most rational choice for imbalanced data, which depicts relative trade-offs between the benefits and costs (Fawcett). ROC curve is generated based on a basic matrix in machine learning called confusion matrix. Table 1 is a confusion matrix of a binary classification problem.

As is shown in Table 1, the classes in binary classification problems are divided into positive class and negative class respectively. ROC curve plots $\frac{FP}{TN+FP}$ (FPR) on the X-axis and plots $\frac{TP}{TP+FN}$ (TPR) on the Y-axis, where different pair of (FPR, TPR)s can be obtained by changing the thresholds of the classifier (Richard and Jonathan, 2006). The threshold of a classifier presents the degree to which an instance is a member of a class (Fawcett, 2006). In practice, we often use the Area Under ROC Curve (AUC) as the scalar measure instead of ROC curve, the larger AUC value is the better. The traditional ROC curve can only be applied to binary classification problems.

Table 1
Confusion matrix of binary classification problem.

	Predicted positive class	Predicted negative class
Positive class	TP	FN
Negative class	FP	TN

There are three extensions for the AUC for multi-class case, OVO approach (Nakas and Yiannoutsos, 2004), OVA (David and Robert, 2001) approach and directly extension (Thomas and Robert, 2006; Ferri et al., 2003). OVO and OVA approaches aim to measure the overall performance by evaluating the performance of each class. Specifically, OVO approaches consider the AUC measure between all pairs of classes (regardless of other classes) while OVA approaches take all the other classes into account when computing each single AUC value. More directly extensions have been studied in Thomas and Robert (2006), Ferri et al. (2003) and Lachiche and Flach (2003), where all of extensions are based on Volume Under the ROC Surface(VUS). Though VUS represent a theoretical justification of the classifiers, it is hard to visualize and compute. For c classes cases, each point of ROC surface lies in an $c \bullet (c-1)$ space, while the complexity of computing one n points convex hull is $o(n^d)$ (Ferri et al., 2003). Researches in Thomas and Robert (2006) and Ferri et al. (2003) provide some methods of estimating VUS through the maximum and minimum VUS, in which a cost matrix is needed. The advantages of OVO and OVA approaches are the natrural intuition and easy computation. In particular, through OVO approach we can clarify the misclassification of pair of classes, which is essential for applications like fault detection.

Here we extend AUC to the multiple classification problems by OVO approach, i.e., for a dataset that contains c classes, we calculate AUC values for each pair of classes respectively. Finally we can get C_c^2 AUC values. In order to get a scalar metric, we need a fusion strategy to integrate all the AUC values as the final result. A natural fusion strategy is to output the average value of all the AUC values as the final result, but this approach has a defect. For different classifiers, each single AUC value may be different, but the mean may remains the same. For example, assuming q AUC values are $r_1, r_2, \ldots r_q$ respectively, when the th AUC value and the mth AUC value change into $r_n - \delta$ and $r_m + \delta$ at the same time, the average AUC after changing is $AUC_{avg} = r_1 + r_2 + \ldots + (r_m + \delta) + (r_n - \delta) + \ldots + r_q = \sum_{i=1}^{q} r_i$, which remains to be the same as before, thus we cannot estimate which hypothesis is better. In this paper, we choose an effective method of fusing single OVO AUC values proposed in (Hassan et al., 2010), the idea is as follows: all the AUC values are plotted in a polar coordinate, we calculate the area covered by the polar diagram as the final evaluation metric, the larger the area is, the better. This measure is named as *AUCarea*. The AUC polar diagram for a three-classes problem is shown in Fig. 1.

The formula to compute AUCarea is described as Eq. (1) (the detail analysis can be found in Hassan et al. (2010)):

$$\text{AUCarea} = \frac{1}{2} \sin\left(\frac{2\pi}{q}\right)\left(\left(\sum_{t=1}^{q-1} r_i \times r_{i+1}\right) + (r_q \times r_1)\right) \quad (1)$$

Here $q = C_c^2$ (c is the number of classes), r_i is the ith AUC value.

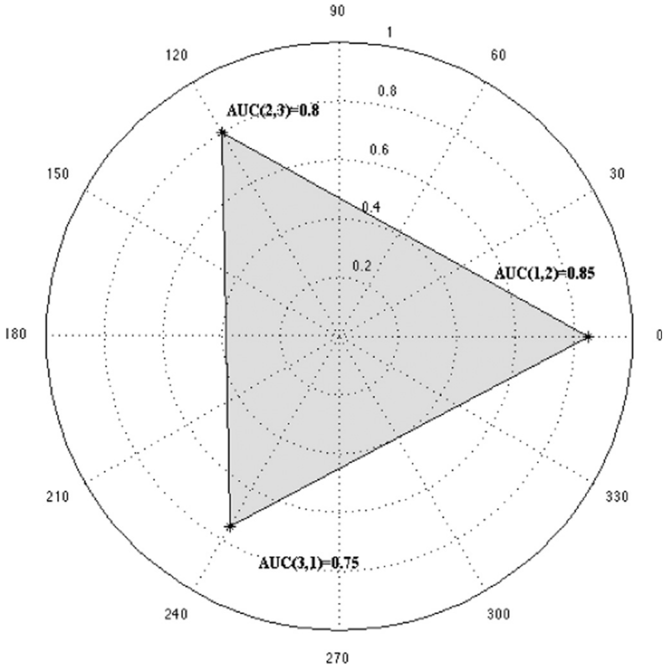

Fig. 1. Polar diagram of three-classes case ($AUC_{1,2}=0.85$, $AUC_{2,3}=0.8$, $AUC_{1,3}=0.75$).

Considering most of metrics are ranged within [0,1], we normalized *AUCarea* by AUCarea/Maximum AUCarea, where Maximum AUCarea is calculated by Eq. (1) when setting all r_is (all AUV values) to 1. The normalized AUCarea can be obtained by Eq. (2). In the following paper, we will use normalized AUCarea instead of AUCarea (but we simply call normalized AUCarea as AUCarea).

$$\text{AUCarea} = \frac{\frac{1}{2}\sin\left(\frac{2\pi}{q}\right)\left(\left(\sum_{t=1}^{q-1} r_i \times r_{i+1}\right) + (r_q \times r_1)\right)}{\frac{1}{2}\sin\left(\frac{2\pi}{q}\right)\bullet q} = \frac{\left(\sum_{i=1}^{q-1} r_i \times r_{i+1}\right) + (r_q \times r_1)}{q} \tag{2}$$

Another reason we believe AUCarea is better than average AUC is that AUCarea is more sensitive for those pairs of classes that gain poor AUC values. In other words, it turns out that if there exists very poor AUC, AUCarea will be poor too. For this part, the reader can check the detail analysis in 3.2 where we put further discussion of AUCarea and average AUC based on our experiments.

Since ROC curve requires the learning models return a probability/score for the prediction of each sample (Fawcett, 2006), that is, the classifiers need to output a C-dimensional probability vector for each sample when dealing with *C*-class problems. Each value in the probability vector stands for the probabilities of a sample that belong to the corresponding class. So in our model, we create two probability matrixes, one for Adaboost and the other one for KNN. Suppose a dataset S that contains D features, m samples and C classes, $S=\{s_1, s_2, \ldots s_m\}$, the class labels are $c_1, c_2, \ldots c_k$, then the probability matrixes of Adaboost and KNN are $m \times C$ matrixes. We denote the probability matrixes of KNN and Adaboost as $Knn\_score_{m\times C}$ and $Ada\_score_{m\times C}$ respectively. Here $Knn\_score(i,j)$ and $Ada\_score(i,j)$ are the probabilities of sample s_i that is assigned to jth class by the KNN and Adaboost-KNN respectively.

For sample s_i, we record all the labels of the k nearest neighbors of s_i. Let n_j represents the number of s_i's neighbors that belongs to jth class, $\sum_{j=1}^{C} n_j = k$, then $Knn\_score(i,j)$ can be computed by setting $Knn\_score(i,j) = n_j/k$. Assuming the number of iterations of Adaboost algorithm is T, in each iteration an error rate ε_t and $Knn\_score_t(i,j)$ are computed. The corresponding $Ada\_score(i,j)$ can be defined as the average of weighted score according to to ε_t and $Knn\_score_t(i,j)$, that is,

$$Ada\_score(i,j) = \sum_{t=1}^{T}\left(\frac{1-\varepsilon_t}{\sum_{t=1}^{T} 1-\varepsilon_t} \times Knn\_score_t(i,j)\right) \tag{3}$$

where $\frac{1-\varepsilon_t}{\sum_{t=1}^{T} 1-\varepsilon_t}$ is the weight of tth KNN.

The way for computing *AUCarea* after calculating *Ada_score* is as follows: for class a and class b, we choose the class that contains relative more samples as the positive class. When computing $AUC_{a,b}$, those samples that Adaboost returns 0 probabilities both for class a and class b are ignored. That is, for sample i, if $Ada\_score(i,a) = Ada\_score(i,b) = 0$, then sample i will not be considered when computing $AUC_{a,b}$. after all the AUCs are computed, the corresponding *AUCarea* can be computed by Eq. (2).

2.3. Binary PSO for feature selection

Samples of different classes may overlap, which makes classifier hard to find the boundaries among different classes. When facing imbalanced data, the samples of minority class are rare so that they may be treated as noise easily. In that case, if the boundaries of different classes are ambiguous, the classifier may fail to detect the structure of minority class samples. Feature selection is a powerful method to solve this problem. We choose the logging data which we use in Section 4 to illustrate this observation. We select three features of logging data and plot the sample distribution, which is shown on the right side of Fig. 2, where the diamonds are the oil layer samples and the circles are non-oil layer samples. It is clear that the boundary between these two classes is ambiguous. On the left side of Fig. 2 we present a 2-D sample distribution by removing one of the three features (CNL). It can be observed that the boundary is much more easy to detect.

While employing feature selection to imbalanced data classification, a main challenge is the trade-off between removing the irrelevant features and keeping useful features. Similar to resampling technologies, remove somewhat irrelevant features may also a risk of losing potentially useful information because the original data distribution may be altered by feature selection procedure (Sun et al., 2015). For this reason, instead of utilizing filter methods such as algorithm used in Krawczyk and Schaefer (2013), we would rather to choose wrapper methods, since it is hard to judge the correlation between features and targets. Addressing feature selection through a meta-heuristic method is one of a worthwhile way for feature selection. Particle Swarm Optimization (PSO) is a widely used stochastic evolutionary algorithm for solving optimization problems, which is devised by Kennedy and Eberhart (1995). Unlike other evolutionary algorithms (such as GA, DE, etc.), PSO is pretty simple and does not contain crossover and mutation operations, which can help us reduce the complexity of our model. Basic PSO is proposed as an optimization technique applied in real space (Bin et al., 2012), while Kennedy and Eberhart also proposed Binary Particle Swarm Optimization (BPSO) algorithm (Kennedy and Eberhart, 1997) that extending PSO to binary space case.

Applying BPSO to implement feature selection, each bit of the position vector of particle indicates whether a feature is selected or not. Generally the fitness of each particle is the classification accuracy based on the subsets selected by BPSO (Inbarani et al., 2014), here we use *AUCarea* as the fitness value. The main

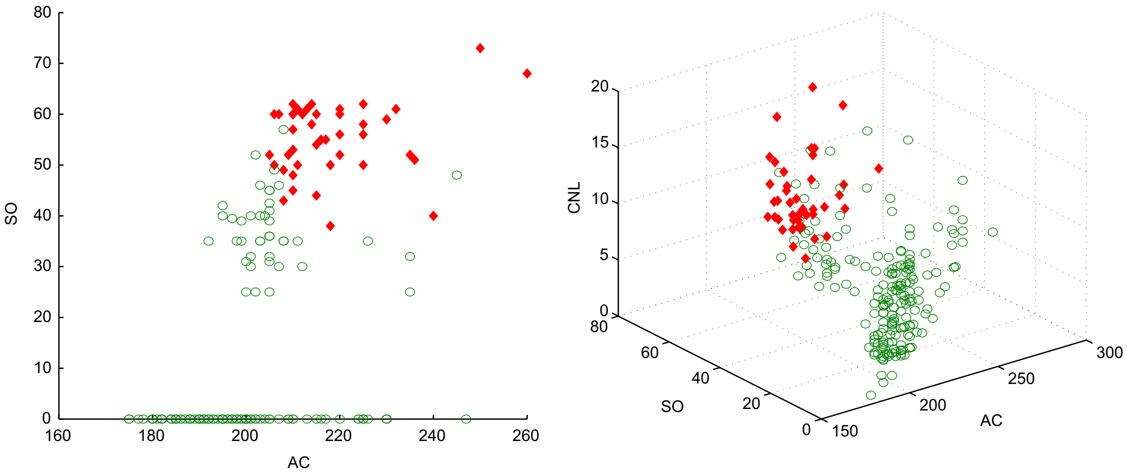

Fig. 2. Sample distribution before and after removing irrelevant feature.

processes of BPSO feature selection algorithm can be found in Supplementary material section.

2.4. BPSO-Adaboost-KNN Ensemble algorithm

In BPSO-Adaboost-KNN ensemble algorithm, we firstly initialize particles in BPSO, each particle stands for a selection of feature set. After initialization, BPSO is conducted to search for best subset of all features. In each iteration of BPSO, Adaboost-KNN is employed to classify the subset of features selected by each particle. When training the Adaboost-KNN classifier, each KNN firstly returns a probability matrix *Knn_score*, then after T times training, *Ada_score* is computed according to all the T *Knn_scores*. Finally we compute *AUCarea* that is described in 2.2 based on *Ada_score* and assign *AUCarea* as the fitness value of each particle in BPSO. The main procedures of BPSO-Adaboost-KNN are given in Algorithm 3. Fig. 3 shows the framework of the whole algorithm.

Algorithm 3. BPSO-AdaBoost-KNN Algorithm

```
Begin
Randomly initialize each particle
While (number of iterations, or the stopping criterion is not met)
  For $i=1$ to $N$ (Population size)
    Selected the subset $Data_i$ based on particle $i$
    Initialize distribution: $D_1(p)\leftarrow 1/M$ (M is the size of training set)
      For $t=1$ to $T$
        Train KNN classifier under the distribution $D_t$ and get the hypothesis $h_t : data_t \to \{c_1, c_2, \ldots c_k\}$, compute probability matrix $Knn\_score_t$
        Calculate the error item for $h_t$:
          $\varepsilon_t \leftarrow \sum_{i=1}^{N} D(i)[y_i \neq h_t(x_i)]$
      Set $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$
      Update the distribution on training set $D_{t+1}(p) = D_t(p)\beta_t^{1-[y_p \neq h_p(x_p)]}$ for all $p \in \{1, 2, \ldots M\}$
    End for $t$
    Evaluate the final hypothesis:
      $H(x) = \operatorname{argmax}_{y \in Y}(\sum_{t=1}^{T} \ln\left(\frac{1}{\beta_t}\right)[y = h_t(x)])$
  Calculate the probability matrix of Adaboost:
      $Ada\_score = \sum_{t=1}^{T}\left(\frac{1-\varepsilon_t}{\sum_{t=1}^{T} 1-\varepsilon_t} \bullet Knn\_score_t\right)$
  Combine all classes into pairs to evaluate $AUCarea_i$, $fitness(x_i) = AUCarea_i$
  Find $pbest_i$ (personal best) for each particle and $gbest$ (global best) in this iteration based on fitness.
  Update $v_{ij}$ and $x_{ij}$ based on standard update formula of BPSO
End while
Output: Best feature subset $s_{best}$, classification result, best AUC polar diagram, all AUCs and $AUCarea_{best}$
```

3. Experiments and analysis

In this section we first introduce the experimental setup and the datasets we used in Section 3.1. Afterwards, we provide details of the comparisons and analyses in Section 3.2. Lastly, statistical studies are carried out in Section3.3.

3.1. Experimental setup and datasets

15 KEEL datasets repository for multiple class imbalanced problems at http://www.keel.es/datasets.php and 4 additional UCI datasets (Bache and Lichman, 2013) are used to test the performance of the proposed algorithm. The detail information about these datasets is summarized in Table 2, where IR is the ratio between the number of minority class sample and the number of majority class sample. In Table 2 we also list the number of selected feature by BPSO. Besides, the second column corresponding to the short name for some datasets, we will use these short names in the rest of this paper.

All experimental studies were conducted by employing 10-fold cross validation (including Section 4). The relevant parameters in BPSO are the same as the corresponding parameters set in Kennedy and Eberhart (1997), where inertia weight is set to 0.729, learning factor c_1, c_2 are both set to 1.49445. In addition, we set k in KNN equals to 5 based on our pre-experiments.

3.2. Experimental results and analyses

In this section, we carried out bunch of experiments to test the performance of the proposed algorithm. The present experiments consist of three investigations. The first investigation is to explore whether both feature selection and boosting are able to improve the performance of our model. Thus, we made comparison among the ensemble algorithm BPSO-Adaboost-KNN and other single algorithms that are used in the ensemble model. These single algorithms are

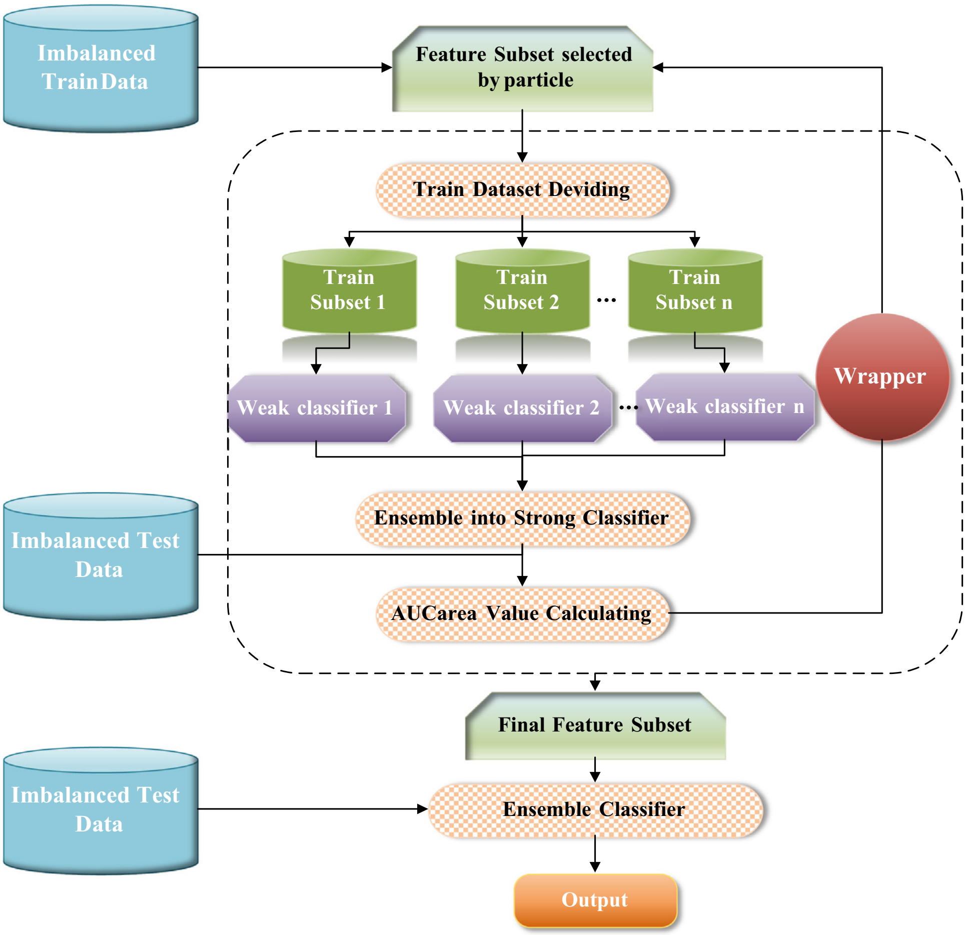

Fig. 3. Ensemble strategy of our model.

Table 2
Detailed information of test datasets.

Dataset	Short name	# Samples	# Classes	Class distribution	IR	# Features	# Selected features
Balance	–	625	3	(49/288/288)	1:6	4	3
Glass	–	214	6	(70/76/17/13/9/29)	1:8	9	6
Landsat	–	2000	6	(461/224/397/211/237/410)	1:2	36	24
New_thyroid	NewT	215	3	(150/35/30)	1:5	4	2
Zoo	–	101	7	(41/20/5/13/4/8/9)	1:10	16	9
Ecoli	–	358	8	(143/77/2/2/35/20/5/42)	1:72	7	5
Hayes-Roth	HayesR	160	3	(65/64/31)	1:2	5	3
Page-Blocks	PageB	5472	5	(4913/329/87/115/28)	1:175	10	7
Wine-Quality-White	WineQW	4898	5	(2198/1457/880/175/163)	1:14	11	5
Splice	–	3190	4	(767/768/1655)	1:2	60	30
Autos	–	159	6	(2/14/33/32/20/9)	1:16	25	8
Contraceptive	Contr.	1473	3	(440/233/357)	1:2	9	4
Dermatology	Derm.	366	6	(77/42/49/33/33/14)	1:6	34	22
Lymphography	Lym.	148	4	(2/42/56/1)	1:41	18	10
Penbased	Pen.	1100	10	(80/79/79/74/79/41/73/80/73/74)	1:2	16	11
Shuttle	–	2175	7	(1194/1/4/236/86)	1:853	9	6
Thyroid	–	720	3	(11/25/466)	1:37	21	6
Wine	–	178	3	(41/49/33)	1:2	13	9
Yeast	–	1484	10	(324/3/24/30/35/114/170/300/14/21)	1:23	8	7

KNN, BPSO-KNN and Adaboost-KNN. Our goal for this comparison lies in demonstrating the effectiveness of each sub-procedure of BPSO-Adaboost-KNN (a deeper statistical test is present in Section 3.3). The second thing we care about is that if proposed algorithm is comparable with previous methods (especially for ensemble methods) that are well accepted. For this part we compared our algorithm with other four state-of-the-art algorithms that used ensemble strategies to deal with imbalanced classification. The last investigation is about AUCarea, since AUCarea is a new metric that we have not found any previous study used it for imbalanced data classification, we want to illustrate its validity and effectiveness.

3.2.1. *Effectiveness of each component of BPSO-Adaboost-KNN*

In order to verify the effectiveness of feature selection and Adaboost that are used in our model, we conducted experiments to compare the BPSO-Adaboost-KNN with simple KNN, BPSO-KNN and Adaboost-KNN. We evaluated the final results by various metrics including AUCarea, total accuracy (denoted as Total acc), accuracy on majority class (denoted as Maj. acc), accuracy on minority class (denoted as Min. acc), average G-means (denoted as GM) and average *F*-value (denoted as FV) (Galar et al., 2012). Table 3 shows the total accuracy, accuracy on majority class and accuracy on minority class of BPSO-Adaboost-KNN and other three separated methods, Table 4 shows the AUCarea, average G-mean and average *F*-value of those four algorithms.

Overall, the results shown in Tables 3 and 4 demonstrate BPSO-Adaboost-KNN far outperforms other three algorithms. While comparing Adaboost-KNN with basic KNN algorithm, Adaboost-KNN achieved superior performance results. In addition, it is obviously that the two algorithms that implement feature selection are better than the corresponding algorithms without feature selection in terms of both total accuracy and the accuracy of minority class, which means BPSO-Adaboost-KNN is better than Adaboost-KNN and BPSO-KNN outperform KNN. These results confirm our initial observation that BPSO eliminated the unrelated or redundant features, so that it can improve the both total accuracy and the accuracy of minority class of our model.

There is another interesting finding for this investigation. While it is confirmed that both BPSO and Adaboost have positive contributions, it is hard to distinguish whose contribution is more significant. For most of datasets in Table 3, both BPSO-KNN and Adaboost-KNN can increase the accuracy for both majority class and minority class, thus lead to a higher total accuracy for entire datasets. However, for those values in bold in Table 3, BPSO-KNN and Adaboost-KNN just obtain a higher accuracy on either majority class or minority class separately. Taking dataset HayesR for example. Compared with KNN, Adaboost-KNN improves the accuracy just on minority class while the accuracy on majority class remains the same. On the other hand, BPSO-KNN significantly increases the performance on majority class, while just gains a slight improvement for minority class.

Overall, from Tables 3 and 4 we can demonstrate that both BPSO and boosting have positive contributions for our model,

Table 3
Comparison of accuracy rate obtained via separated methods in our algorithm.

Dataset	KNN			Adaboost-KNN			BPSO-KNN			BPSO-Adaboost- KNN		
	Total acc	Maj acc	Min acc	Total acc	Maj acc	Min acc	Total acc	Maj acc	Min acc	Total acc	Maj acc	Min acc
Balance	0.5873	0.6000	0.3849	0.7421	0.6000	0.5862	0.7698	0.7759	0.6706	0.9563	0.9000	0.8190
Glass	0.6477	0.6129	1.0000	0.9773	0.8387	1.0000	0.6705	0.6452	1.0000	0.9886	0.8710	1.0000
Landsat	0.7519	0.8353	0.7340	0.9239	0.8617	0.8138	0.8030	0.8670	0.7494	0.9551	0.8794	0.9529
NewT	0.9186	0.9667	0.8121	0.9884	0.9883	0.9133	0.9186	1.0000	0.9133	1.0000	1.0000	1.0000
Zoo	0.7907	0.8824	0.9545	0.9302	**1.0000**	**1.0000**	0.8140	**0.8824**	**1.0000**	1.0000	**1.0000**	**1.0000**
Ecoli	0.8222	0.9649	0.6126	0.9111	1.0000	0.7637	0.8296	0.9649	0.7521	0.9556	1.0000	0.8519
HayesR	0.5556	0.6190	0.4121	0.6667	**0.6190**	**1.0000**	0.7593	**1.0000**	**0.5556**	0.9444	**1.0000**	**1.0000**
PageB	0.9316	0.9496	0.9188	0.9936	0.9959	1.0000	0.9462	0.9532	0.9323	0.9954	1.0000	1.0000
WineQW	0.5250	0.5011	0.3750	0.8909	0.9614	0.7500	0.5712	0.5432	0.6250	0.8962	0.9750	1.0000
Splice	0.5806	0.5776	0.5076	0.7382	0.7067	0.5306	0.6584	0.5776	0.6034	0.8005	0.7955	0.7830
Autos	0.5694	0.6667	0.3292	0.8388	**0.7267**	**1.0000**	0.6714	**0.8667**	**0.5333**	0.9796	**0.9933**	**1.0000**
Contr.	0.5695	0.7566	0.5400	0.8142	**0.7566**	**0.9800**	0.7876	**0.8942**	**0.5900**	0.9368	**0.9947**	**0.9900**
Derm.	0.8091	0.9545	0.8706	0.9636	1.0000	0.9583	0.9091	0.9818	0.8706	0.9727	1.0000	0.9629
Lym.	0.7872	0.8400	0.6786	0.9087	0.9200	1.0000	0.8298	0.8800	1.0000	0.9210	1.0000	1.0000
Pen.	0.9003	0.9214	0.9688	0.9670	1.0000	1.0000	0.9352	0.9714	1.0000	0.9821	1.0000	1.0000
Shuttle	0.9193	0.9161	1.0000	0.9285	0.9161	1.0000	0.9754	1.0000	1.0000	0.9985	1.0000	1.0000
Thyroid	0.9174	1.0000	0.7696	0.9654	1.0000	1.0000	0.9335	1.0000	0.8993	0.9725	1.0000	1.0000
Wine	0.7636	0.8182	0.8000	0.9218	1.0000	1.0000	0.7818	0.8636	0.8000	0.9536	1.0000	1.0000
Yeast	0.5457	0.6187	0.5000	0.7194	0.7171	1.0000	0.5590	0.6187	0.5000	0.7283	0.7911	1.0000

Table 4
Comparison of different metrics obtained via separated methods in our algorithm.

Dataset	KNN			Adaboost-KNN			BPSO-KNN			Adaboost-BPSO-KNN		
	AUC area	GM	FV	AUC area	GM	FV	AUC area	GM	FV	AUC area	GM	FV
Balance	0.5216	0.3647	0.7284	0.5838	0.8490	0.8885	0.6332	0.7488	0.8704	0.7875	0.9891	0.9904
Glass	0.7896	0.6007	0.9042	0.8345	0.9513	0.9898	0.7155	0.6268	0.9180	0.8721	0.9917	0.9943
Landsat	0.7016	0.8562	0.8588	0.8412	0.9577	0.9871	0.7726	0.8782	0.8950	0.8930	0.9632	0.9887
NewT	0.7860	0.8635	0.9348	0.9172	0.9814	0.9912	0.8768	0.9407	0.9365	1.0000	1.0000	1.0000
Zoo	0.8362	0.7567	0.9759	0.9822	0.9998	0.9999	0.9431	0.8479	0.9659	1.0000	1.0000	1.0000
Ecoli	0.7398	0.8532	0.9707	0.8858	0.9232	0.9887	0.8137	0.8723	0.9723	0.8996	0.8506	0.9933
HayesR	0.4697	0.4695	0.6529	0.5448	0.9474	0.9714	0.6279	0.7966	0.8665	0.7589	0.9807	0.9910
PageB	0.8057	0.7919	0.9350	0.9364	0.9552	0.9945	0.9066	0.9093	0.9448	0.9465	0.9419	0.9940
WineQW	0.6520	0.6531	0.7269	0.7854	0.7623	0.8782	0.6893	0.6828	0.8490	0.8454	0.7531	0.8782
Splice	0.4795	0.3910	0.6480	0.6253	0.6198	0.7463	0.5468	0.4959	0.7846	0.6943	0.7348	0.7846
Autos	0.6046	0.5356	0.8816	0.8096	0.9021	0.9796	0.7443	0.6283	0.9260	0.9270	0.9451	0.9802
Contr.	0.4480	0.4520	0.4520	0.6055	0.5268	0.6739	0.5867	0.4752	0.6939	0.6523	0.5303	0.7164
Derm.	0.8099	0.9306	0.9817	0.9245	0.9450	0.9821	0.9063	0.8388	0.9618	0.9403	0.9646	0.9919
Lym.	0.7287	0.4106	0.9040	0.8229	0.8088	0.9192	0.7615	0.6332	0.9256	0.8617	0.9391	0.9623
Pen.	0.9031	0.8664	0.9233	0.9773	0.9883	0.9997	0.9221	0.9750	0.9950	0.9952	0.9974	0.9998
Shuttle	0.9114	0.9090	0.9890	0.9244	0.9210	0.9410	0.9691	0.9260	0.9990	0.9890	0.9407	0.9996
Thyroid	0.5415	0.5943	0.6869	0.5668	0.6007	0.7786	0.8147	0.8310	0.9286	0.9286	0.9262	0.9979
Wine	0.8340	0.8135	0.8929	0.8948	0.8140	0.8140	0.8974	0.8135	0.8929	0.9064	0.8141	0.8932
Yeast	0.4593	0.5610	0.7107	0.6055	0.6539	0.8853	0.5499	0.5610	0.7390	0.6649	0.6614	0.9386

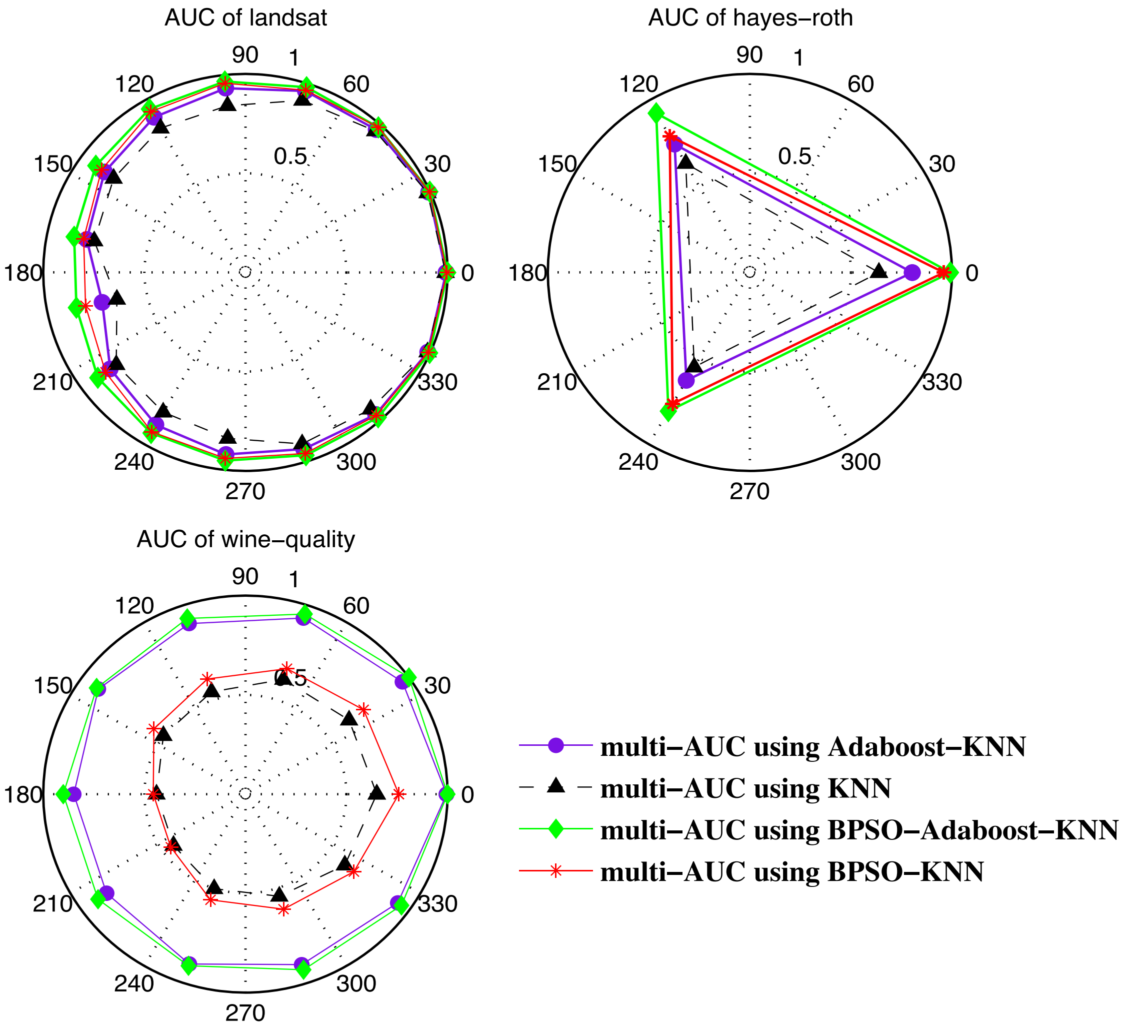

Fig. 4. Comparison of AUC polar of benchmark datasets.

especially for classifying the minority class samples which we care more about in imbalanced data (see Table 3).

At last, Fig. 4 presents the AUC polar diagrams of four algorithms mentioned above. We can evaluate the performance of all four algorithms intuitively through the area shown in polar diagrams (for the limit of space here we just show the diagrams of three benchmarks), the larger area is the better.

3.2.2. *Comparison among BPSO-Adaboost-KNN and state-of-the-art algorithms*

In this comparison we choose four efficient ensemble algorithms proposed previously, i.e., SMOTEBoost (Nitesh et al., 2003), EasyEnsemble (Liu et al., 2009), PUSBE (Krawczyk and Schaefer, 2013), a cost-sensitive decision tree ensemble algorithm (Krawczyk et al., 2014), and a novel ensemble algorithm that using Kmeans to split majority samples (Sun et al., 2015).

SMOTEBoost and EasyEnsemble are two popular ensemble algorithms that have been used for comparison in dozens of previous papers (Galar et al., 2013; López et al., 2013; Sun et al., 2015; Krawczyk et al., 2014; Krawczyk and Schaefer, 2013). Both of these two methods are based on sampling and boosting technologies, in which SMOTEBoost use SMOTE for over-sampling minority class while EasyEnsemble employ random under-sampling method on majority class. PUSBE is a novel ensemble algorithm that a filter method for feature selection is carried out. Another contribution of PUSBE lies in the utilization of double-fault diversity measure for pruning basic classifiers. It is interesting to take PUSBE into comparison since it also considered feature selection in the ensemble model. The proposed ensemble algorithm in Krawczyk et al. (2014) is based on a cost-sensitive basic classifier and also uses stochastic evolutionary algorithm to fusion basic classifiers. ClusterBal uses cluster method to split majority samples, converts an imbalanced dataset into multiple balanced sub-datasets and then trains multiple base classifiers. Finally, the classification results of all the base classifiers are combined by a specific ensemble rule called MaxDistance. All of these algorithms select different base classifiers, such as SVM, Ripper, CART, etc. (we choose KNN as basic classifier in ClusterBal). Note that all of these five algorithms have only been used in binary classification problems in the correspongding papers, we generalizing these methods into multiple case by using OVO approach. The brief descriptions of these four algorithms are shown in Table 5. Table 6 presents the performance of BPSO-Adaboost-KNN and these four algorithms in terms of AUCarea, average G-mean and average *F*-value, where CS-MCS in the last column stands for the cost-sensitive decision tree ensemble algorithm proposed in Krawczyk et al. (2014).

The superior algorithm in terms of AUCarea, average G-mean and average *F*-value are highlighted in bold in Table 6. The results show that all the five algorithms are comparable, since all of them have obtained the best performance in different datasets with respect to the others. Overall, PUSBE and BPSO-Adaboost-KNN are slightly better than the other algorithms, while BPSO-Adaboost-KNN is more universal for imbalanced datasets with different imbalance ratio. For example, BPSO-Adaboost-KNN perform well in those datasets whose imbalance ratios are below 1:10 (such as Balance, Glass, etc.), as well as in those datasets which are extremely imbalanced(such as Autos and Shuttle). Since PUSBE is also feature selection based ensemble algorithm, this finding can illustrate the crucial role of feature selection for imbalanced data. Another observation is that under-sample based methods like EasyEnsemble, CS-MCS and ClusterBal(we treat ClusterBal as under-sample based method for the reason that this algorithm uses Kmeans to split majority samples into each sub-dataset, in some sense it is an under-sample approach) perform bad in benchmarks that are highly imbalanced, such as Zoo, PageB and Shuttle. This may because under-sample methods drop too much information of majority classes in order to balance the sub-datasets. Taking ClusterBal for example. In data balancing process, ClusterBal split majority samples into K clusters, where K is set as the ratio of majority class samples over the minority class samples. When dataset is extremely unbalance, say only one sample in the minority class of the training set, then ClusterBal will split the majority class into huge amount of clusters while each cluster contains only one or a few samples. In this way, all the base classifiers cannot be trained well because the sub-training-sets are too small (although they may be balanced). Another reason some state-of-the-art algorithms like ClusterBal failed to achieve ideal results as they were reported in the proposed paper is that boundaries among multiple classes in each balanced sub-dataset become distorted when we just use a small cluster of multiple majority classes to generate balanced datasets.

Table 5
Ensemble strategies of four state-of-the-art algorithms.

Algorithm	Ensemble strategy
SMOTEBoost	SMOTE+Adaboost+Ripper (base classifier)
EasyEnsemble	Random under-sample +Bagging+Adaboost+CART (base classifier)
PUSBE	Filter feature selection+Boosting+SVM(base classifier)+Double-fault diversity based pruning method+Classifier fusion based on BP
CS-MCS	Random under-sample +Cost sensitive decision tree+Classifier fusion based on GA
ClusterBal	Split majority classes by Kmeans+KNN(base classifier)+Max-Distance ensemble rule

We also compare the runtime of all the algorithms used in this section, as is shown in Table 7. Since all the five algorithms except ClusterBal are ensemble algorithms which need to train basic classifiers in a loop, the computational complexity for this five algorithms are relatively expensive. In general ClusterBal wins in most of benchmarks besides some large and highly imbalanced datasets. In particular, BPSO-Adaboost-KNN, PUSBE and CS-MCS employ evolutionary algorithms as a sub-component of the entire model, which lead to a higher computational cost with respect to SMOTEBoost and EasyEnsemble. However, as we can see, the runtime of BPSO-Adaboost-KNN is slightly lower than PUSBE and CS-MCS. The reason for that lies in that we use the simplest basic classifier KNN, while PUSBE and CS-MCS used SVM and decision tree as basic classifiers, which have much higher computational complexity than KNN. We also notice that although no evolutionary algorithm has been employed in SMOTEBoost, the computational cost of SMOTEBoost is much higher than EasyEnsemble, sometimes even higher than the proposed algorithm. The main reason is, the basic classifier used in SMOTEBoost is Ripper, which has a time complexity of $O(N^4)$. Another reason lies in that SMOTE creates new training samples which increases the training space while EasyEnsemble exploits under-sampling method, which just uses a subset of training data to train each basic classifier.

3.2.3. Discussion about the validity and goodness of AUCarea

Considering *AUCarea* is a novel metric, we would also like to find out if *AUCarea* is valid to evaluate performance of learning models just as other widely-used metrics such as Average-AUC, Average G-means, and Average *F*-value (Galar et al., 2012). In order to show the consistence between AUCarea and other valid metrics, Bland–Altman analysis and linear regression are carried out in this section.

Bland–Altman analysis is a common method used to analyze the agreement between different assays in chemistry and biostatistics (Bland and Altman, 1986). Here we use it to prove the consistency of our metric with other valid measures. Bland–Atman plot is a scatter diagram that plots all values of two metrics. In Bland–Atman diagram the mean of two metrics is taken as *x*-axis and the difference between two metrics is taken as *y*-axis. Bland–

Table 6
Compare proposed algorithm with four state-of-the-art algorithms.

Dataset	Adaboost-BPSO-KNN			SMOTEBoost			EasyEnsemble			PUSBE			CS-MCS			ClusterBal		
	AUCarea	GM	FV	AUCarea	GM	FV	AUCarea	GM	FV	AUCarea	GM	FV	AUCarea	GM	FV	AUCarea	GM	FV
Balance	**0.788**	**0.989**	**0.99**	0.679	0.929	0.942	0.774	0.946	0.946	0.629	0.863	0.894	0.651	0.718	0.865	680	0.807	0.791
Glass	**0.872**	**0.992**	**0.994**	0.759	0.955	0.983	0.852	0.986	0.992	0.858	0.899	0.984	0.807	0.865	0.898	0.833	0.899	0.902
Landsat	**0.893**	0.963	**0.989**	0.882	0.959	0.988	0.886	**0.966**	**0.989**	0.778	0.927	0.977	0.758	0.723	0.949	0.882	0.961	0.984
NewT	**1**	**1**	**1**	0.984	0.998	0.999	**1**	**1**	**1**	**1**	**1**	**1**	0.923	0.948	0.965	**1**	**1**	**1**
Zoo	**1**	**1**	**1**	**1**	**1**	**1**	0.894	0.771	0.976	**1**	**1**	**1**	0.831	0.745	0.978	0.881	0.791	0.765
Ecoli	0.9	0.851	**0.993**	0.785	0.733	0.971	0.871	0.721	0.988	0.751	0.776	0.974	**0.935**	**0.876**	0.974	0.821	0.843	0.925
HayesR	0.759	0.981	0.991	0.635	0.707	0.793	**1**	**1**	**1**	0.876	0.891	0.941	0.776	0.829	0.828	**1**	**1**	**1**
PageB	0.847	0.942	0.994	0.828	0.963	0.988	0.878	0.972	0.997	**0.899**	**0.978**	**0.995**	0.79	0.815	0.86	0.401	0.31	0.579
WineQW	**0.845**	0.753	0.878	0.704	**0.861**	0.814	0.814	0.854	0.877	0.78	0.809	**0.926**	0.7	0.799	0.845	0.6	0.664	0.88
Splice	0.694	0.635	0.785	0.616	0.558	0.756	0.697	0.612	0.76	0.692	**0.841**	0.823	0.726	0.722	0.75	**0.767**	0.769	**0.841**
Autos	**0.927**	0.945	**0.98**	0.797	**0.953**	0.882	0.727	0.927	0.871	0.804	0.798	0.943	0.832	0.915	0.969	0.829	0.814	0.893
Contr.	0.652	0.53	0.716	0.542	0.57	0.736	0.55	0.59	0.794	0.537	0.634	0.699	0.493	0.615	0.751	**0.749**	**0.816**	**0.799**
Derm.	0.94	0.965	0.992	0.973	0.958	0.99	0.934	0.972	0.994	**1**	**1**	**1**	0.97	0.973	0.995	0.8	0.845	0.907
Lym.	0.862	0.839	0.962	0.843	0.666	0.913	0.804	0.99	**0.993**	0.873	0.981	0.986	**0.926**	**0.997**	0.982	0.66	0.869	0.856
Pen.	0.995	0.997	0.999	0.997	**0.998**	**0.999**	**0.998**	**0.998**	**0.999**	0.951	0.981	0.996	0.751	0.826	0.969	**0.998**	0.971	0.949
Shuttle	0.989	0.941	**0.999**	**0.996**	0.962	0.994	0.825	0.9	0.988	0.914	0.899	0.999	0.916	**0.981**	0.961	0.6	0.679	0.84
Thyroid	0.929	0.926	0.998	0.813	0.998	0.99	**1**	**1**	**1**	0.879	0.998	0.99	0.925	0.961	0.946	0.835	0.864	0.798
Wine	0.906	0.814	0.893	0.916	0.998	0.901	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Yeast	0.665	0.661	0.939	0.674	0.861	0.987	**0.78**	**0.853**	**0.982**	0.688	0.842	0.975	0.603	0.588	0.941	0.428	0.583	0.944

Table 7
Computational cost for each algorithm (runtime in terms of milliseconds).

Dataset	BPSO-Adaboost-KNN	SMOTEBoost	EasyEnsemble	PUSBE	CS-MCS	ClusterBal
Balance	7496	9220	5042	9624	7430	**737**
Glass	1877	1821	1679	1604	1998	**332**
Landsat	16,877	12,339	**10,982**	18,228	19,601	13,368
NewT	5408	6093	5032	6452	6837	**1276**
Zoo	2652	2617	2162	1868	2415	**268**
Ecoli	**4076**	6755	4083	4883	5167	4917
HayesR	668	214	**201**	652	974	276
PageB	28,941	129,871	**11,075**	26,560	39,250	183,209
WineWQ	24,466	74,187	**15,386**	24,968	32,029	69,678
Splice	10,955	16,429	6912	11,968	11,073	**1566**
Autos	5305	5799	5152	6021	6910	**1340**
Contr.	12,809	9571	**9140**	11,194	23,733	23,434
Derm.	4799	2304	2100	3478	5078	1524
Lym.	**1005**	2663	1197	1228	1122	1488
Pen.	9140	3687	3459	6667	10,291	**1492**
Shuttle	15,960	44,192	**11,750**	13,494	18,310	50,434
Thyroid	8503	8963	7248	8143	9641	**2235**
Yeast	12,026	9410	**7325**	11,849	18,419	10,031
Wine	1243	1116	974	1789	1906	**184**

Altman analysis allows us to investigate the existence of any correlation between two metrics as well as to identify possible outliers. The mean difference is the estimated bias, and the standard deviation (std) of the differences reflects the random fluctuations around this mean. If the mean value of the difference is far away from 0, this indicates the presence of fixed bias. On contract, if it is close to 0, it can prove the consistence of two measurements. Generally, we also compute 95% limits of agreement for each comparison. If two metrics are correlation, most of plots should be located within the agreement interval. The Bland–Altman plots for each pair of metrics are shown in Fig. 5. It is obviously that AUCarea is strongly correlation with AVG–AUC, AVG-Gmean and AVG-Fvalue, since almost all the dots(except for couple of outliers) are located within 95% limits of agreement (the bottom and top line in diagrams), the mean difference line (middle line) is also close to 0. In comparison between accuracy and AUCarea, 77.5% plots are located in limited lines, which indicates that there exists relative correlation but a little bias between AUCarea and accuracy.

As is mentioned previously classification accuracy may ignore the minority class or treat them as noise. Therefore as long as majority class samples are classified perfectly, accuracy would be high. On contrast, the value of AUCarea may worse than accuracy for the same hypothesis. In order to prove that, we use linear regression to find the regression coefficient for each pair of metrics. Linear regression algorithm is a traditional approach for modeling the relationship between two variables (Chawla et al., 2004). We treat one metric as dependent variable and select another metric as the explanatory variable (independent variable). Using linear regression algorithm, we can model the slope between two metrics, which stands for the coefficient of a pair of metrics. The more the coefficient is close to 1, the more similar the two metrics. Specifically, if the coefficient is equal to 1 with high confidence, the two metrics are fungible. The results of linear regression analysis are shown in Table 8, where we take the column in left side of the table as the independent variable. When taking accuracy as independent variable, the regression coefficient between accuracy and other metrics are much less than 1 with low *p*-value, especially for AUCarea, which reflect that high accuracy rate is indeed an illusion in imbalanced data classification. Secondly, the regression coefficients between AUCarea and other three metrics (AVG–AUC and AVG-*F*value) are near but slightly larger than 1, which indicates that AUCarea can be used as metric for imbalanced data classification just like AVG–AUC and AVG-Fvalue.

However, since AUCarea and average AUC are all based on OVO AUC, why AUCarea is often less than AVG–AUC? To find the reason, we can analyze the formulas for both of them. First AVG–AUC can be computed through Eq. (4).

$$\mathrm{AVG}-\mathrm{AUC}=\frac{\sum_{i=1}^{q} r_i}{q} \tag{4}$$

Assuming that r_i is changed to $r_i - l$, then we can compute the relatively change of average AUC and AUCarea: $\Delta\mathrm{AVG}-\mathrm{AUC}=\frac{l}{q}$ and $\Delta\mathrm{AUCarea}=\frac{l\times(r_{i-1}+r_{i+1})}{q}$. Since almost AUC values are larger than 0.5, ΔAUCarea would slightly larger than $\Delta\mathrm{AVG}-\mathrm{AUC}$. This suggests that if any single AUC decreased, AUCarea would decrease more than AVG–AUC. The above analysis illustrates that AUCarea is more sensitive when any pairs of classes gain poor AUC value. Another intuition is that, AUCarea sums the product of a pair of AUC values in the numerator, while AVG–AUC just sums each single AUC value in the numerator. If there exist a poor AUC value, it will affect two terms of the sum in the numerator of AUCarea, but just one term in the numerator of AVG–AUC. For imbalanced data, the AUC values between minority class and some other classes are often poor. For this reason, AUCarea is superior to AVG–AUC.

3.3. Statistical studies

In order to analyze if there exist significant improvements when we added BPSO and Adaboost into ensemble, as well as to compare the performance of the proposed algorithm with other state-of-the art algorithms from the statistical perception, we recalled a series of nonparametric statistical tests (Alberto et al., 2013). First, we used Wilcoxon paired signed-rank test (Wilcoxon, 1945) to do pairwise comparisons, computed the *p*-value in each comparison, which represents the lowest level of significance of a hypothesis that results in a rejection. Next, Freidman test (Sheskin, 2006) was employed to detect differences among overall performances of all the algorithms. After that, we chose post-hoc test to check out if BPSO-Adaboost-KNN is significantly better than the rest (Holm, 1979).

3.3.1. Statistical tests of BPSO-Adaboost-KNN, BPSO-KNN, Adaboost-KNN and KNN

The results of Wilcoxon tests are shown in Table 9. Between the approaches contained BPSO and those without BPSO, the former one (BPSO-KNN and BPSO-Adaboost-KNN) obtain higher ranks.

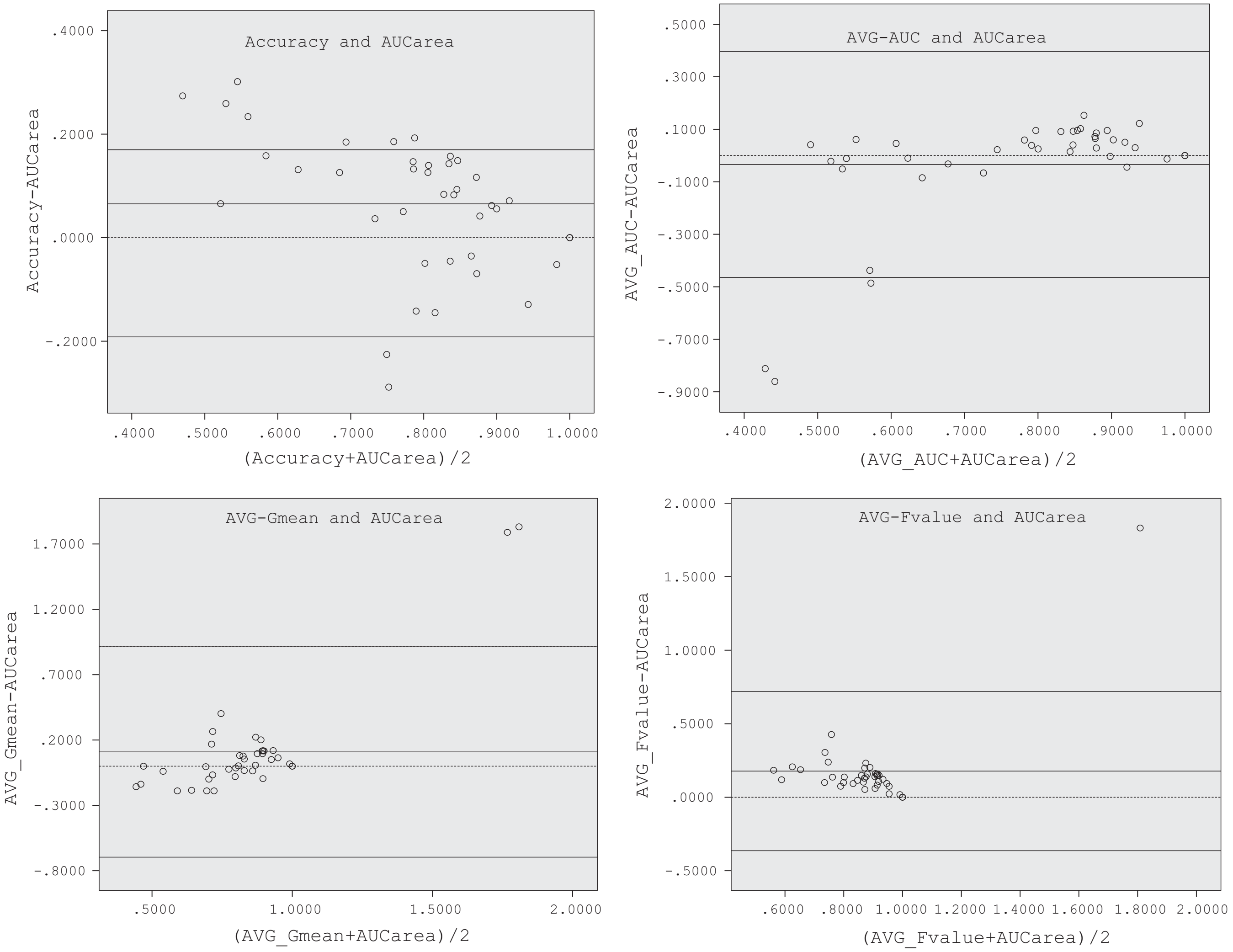

Fig. 5. Bland–Altman diagrams for AUCarea and other metrics.

Table 8
Linear regression analysis of each pair of metrics.

		AUCarea	Accuracy	AVG–AUC	AVG-Gmean	AVG-Fvalue
Regression coefficient	AUCarea	1.000	1.934	1.171	0.678	1.174
	Accuracy	0.517	1.000	0.654	0.665	0.920
	AVG–AUC	0.854	1.529	1.000	0.610	1.311
	AVG-Gmeans	1.474	1.503	1.639	1.000	1.340
	AVG-Fvalue	0.852	1.089	0.763	0.746	1.000
p-value	Nor-AUCarea	.	0.001	0.003	0.006	0.017
($\alpha = 0.05$)	Accuracy	0.001	.	0.011	0.045	0.060
	AVG–AUC	0.003	0.011	.	0.072	0.017
	AVG-Gmeans	0.006	0.045	0.072	.	0.001
	AVG-Fvalue	0.017	0.060	0.017	0.001	.

This result stresses that the feature selection indeed has significant contribution in our model. In addition, Adaboost-KNN and BPSO-Adaboost-KNN win higher ranks with respect to KNN and BPSO-KNN. At last, the rank between Adaboost-KNN and BPSO-KNN is close and *p*-value in this comparison shows low significance, which indicates that the contributions of Adaboost and BPSO are hard to judge.

After pairwise tests, we started the overall comparison by executing the Friedman test which returns a *p*-value of 0.001, as Fig. 6 shows. From Friedman test we can see Adaboost-KNN gains higher rank than BPSO-KNN, which illustrates that boosting contributes more to our ensemble model. The average rank obtained by BPSO-Adaboost-KNN is much higher than the other three methods. Hence, we applied the post-hoc procedure to compare BPSO-Adaboost-KNN with the rest of methods. Observing the results shown in Table 10, BPSO-Adaboost-KNN clearly outperforms the other methods with significant differences.

3.3.2. *Statistical tests of BPSO-Adaboost-KNN and five state-of-the-art algorithms*

The same statistical study processes are carried out for BPSO-Adaboost-KNN and the state-of-the-art algorithms, which are shown in Tables 11 and 12 and Fig. 7. From Table 11 we can observe the ranks of BPSO-Adaboost-KNN are larger than other five algorithms with high significance in comparison of SMOTRBoost, CS-MCS and ClusterBal. In Friedman test, BPSO-Adaboost-KNN gains largest average rank, but the ranks of EasyEnsemble and PUSBE are also comparable. In Table 12, we tested if there exist significant differences between BPSO-Adaboost-KNN and other five algorithms. The results show only SMOTEBoost, ClusterBal and CS-MCS are significantly poorer than BPSO-Adaboost-KNN, while EasyEnsemble and PUSBE is comparable.

Table 9
Wilcoxon tests for BPSO-Adaboost-KNN and the component algorithms.

Comparison	R^+	R^-	Hypothesis ($\alpha = 0.05$)	*p*-Value	Selected
Adaboost-KNN vs. KNN	190.00	0.00	Adaboost-KNN > KNN	0.000	Adaboost-KNN
BPSO-KNN vs. KNN	181.00	9.00	BPSO-KNN > KNN	0.000	BPSO-KNN
Adaboost-KNN vs. BPSO-KNN	139.00	51.00	BPSO-KNN > Adaboost-KNN	0.064	Adaboost-KNN
BPSO-Adaboost-KNN vs. Adaboost-KNN	180.00	10.00	BPSO-Adaboost-KNN > Adaboost-KNN	0.000	BPSO-Adaboost-KNN
BPSO-Adaboost-KNN vs. BPSO-KNN	190.00	190.00	BPSO-Adaboost-KNN > BPSO-KNN	0.001	BPSO-Adaboost-KNN

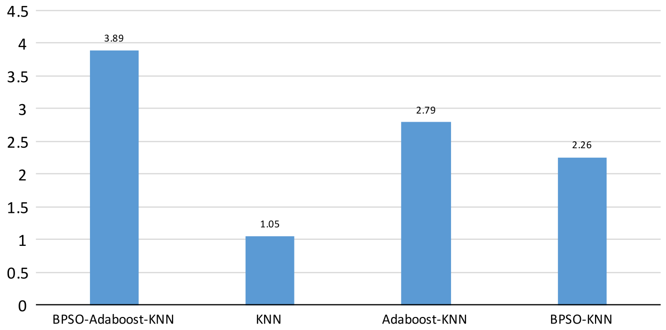

Fig. 6. Average rankings of BPSO-Adaboost-KNN and the component algorithms.

Table 10
Post hoc tests of the component algorithms (taking BPSO-Adaboost-KNN as control method).

Algorithm(Rank)	*Z*	*p*-Value	Hypothesis($\alpha = 0.05$)
Adaboost-KNN(2.5)	3.218	0.001	Rejected for BPSO-Adaboost-KNN
BPSO-KNN(2.4)	3.622	0.001	Rejected for BPSO-Adaboost-KNN
KNN(1.1)	3.823	0.000	Rejected for BPSO-Adaboost-KNN

4. The application of BPSO-Adaboost-KNN algorithm in Oil-bearing of reservoir recognition

In this section, we apply our model in Oil-bearing of reservoir recognition. The descriptions of logging datasets are shown in Section 4.1, the experimental results and analyses are described in Section 4.2.

4.1. Data description

The Oil-bearing of reservoir recognition data has 6 logging attributes, including: Acoustic travel time(AC), Compensated Neutron Logging(CNL), Pesistivity(RT), Porosity(POR), Oil Saturation(SO) and Permeability(PERM). AC can be obtained by acoustic logging equipment, analyzing the property that the sonic propagation varies when it comes to different rocks and fluids. Generally, AC would increase dramatically if there were oil vapor in the void. Various effects of interaction between CNL and other substances can be used to study rock formation properties of the cross section. RT is a physical quantity that represents resistance properties of various substances, as well as a main parameter that judges fluids properties of reservoir. Commonly, resistance increases if there is oil vapor while decreases if there is water in reservoir. POR is defined as the ratio of the void space in a rock to the bulk volume of that rock multiplied by 100 to express in percent. SO is defined as ratio of void volume occupied by crude oil to total void volume of rock in oil reservoir. Allowable capability of fluid passing to the rock in some difference of pressure is called PERM, whose measurement is the rate of permeation. The recognizing oil-bearing formation means to recognize the characters of each layer in the well. These characters include oil layer, inferior oil layer, water layer and dry layer. The experimental data Oilsk81, Oilsk82, Oilsk83, Oilsk84, Oilsk85 come from Jianghan oil field of China. Here we chose Oilsk81 well data as the training set. For testing our model, we took Oilsk82, Oilsk83, Oilsk84, Oilsk85 as test set separately, then merged these four test sets into a collection set and test the model again. Table 13 shows a portion of Oilsk81 well logging data and the corresponding conclusions of logging explanations, the sample distributions of 5 logging datasets are shown in Table 14.

4.2. Results and analysis

The optimal feature sets selected by BPSO are shown in Table 15. We can see that the optimal feature set of each well logging is not exactly the same, which reflects the key attributes of logging data are unclear. AC, POR, SO and PERM all have been selected in different test sets, in which AC, POR and SO have been selected more frequently and also have been selected in the collection set, so we can conclude that AC, POR and SO are key reference attributes of the logging data.

Table 16 presents the classification results obtained in 5 test data in terms of accuracy and AUCarea. Fig. 8 is the error curve of 50 times experiments, and Fig. 9 is the AUC polar diagram of five logging datasets.

According to Table 16, the metric ranking from high to low of four algorithms are: BPSO-Adaboost-KNN, Adaboost-KNN, BPSO-KNN, KNN, in which BPSO-Adaboost-KNN achieves more than 98% accuracy. In addition, it can be found in Fig. 8 that Adabost-KNN is a kind of unstable classifier, the classification results of each time test may be different, but the stability of BPSO-Adaboost-KNN is much better than the Adaboost-KNN, it can be interpreted as the feature selection excludes noisy so that it can increase the stability of classifier.

Using AUCarea as evaluation metric, we can also clearly justify the performances of algorithms between any two classes. If the AUC of two classes is 1, it demonstrates that there is no mutual misclassification exists between these two classes, which also suggests there exists large margin between these two classes. The AUC less than 1 means that the classifier makes some mistakes when distinguishing a pair of classes, the closer the AUC value to 1, the lower misclassified cases occur. In Table 17 we present the AUC values of any two classes when BPSO-Adaboost-KNN and KNN are implemented.

It is obvious that most misclassifications happen between oil layer and inferior oil layer as is shown in Table 17. What's more, water layer samples are also misclassified in some case, since the number of water layer sample is extremely rare. However, AUC values of any pairs of layers obtained by BPSO-Adaboost-KNN are significantly better than those obtained by KNN, especially when distinguishing oil layer samples from other layers samples (as is shown in bold in Table 17). In particle, what the oil companies care

Table 11
Wilcoxon tests for BPSO-Adaboost-KNN and the-state-of-the-arts.

Comparison	R^+	R^-	Hypothesis ($\alpha = 0.05$)	p-Value	Selected
BPSO-Adaboost-KNN vs. SMOTEBoost	152.00	19.00	SMOTEBoost > BPSO-Adaboost-KNN	0.004	BPSO-Adaboost-KNN
BPSO-Adaboost-KNN vs. EasyEnsemble	103.5.00	67.5	EasyEnsemble > BPSO-Adaboost-KNN	0.433	BPSO-Adaboost-KNN
BPSO-Adaboost-KNN vs. PUSBE	107.00	46.00	PUSBE > BPSO-Adaboost-KNN	0.149	BPSO-Adaboost-KNN
BPSO-Adaboost-KNN vs. CS-MCS	156.00	34.00	CS-MCS > BPSO-Adaboost-KNN	0.014	BPSO-Adaboost-KNN
BPSO-Adaboost-KNN vs. ClusterBal	136.50	34.50	ClusterBal > BPSO-Adaboost-KNN	0.026	BPSO-Adaboost-KNN

Table 12
Post hoc tests of the-state-of-the-arts (taking BPSO-Adaboost-KNN as control method).

Algorithm(Rank)	Z	p-Value	Hypothesis($\alpha = 0.05$)
SMOTEBoost(2.47)	2.896	0.001	Rejected for BPSO-Adaboost-KNN
EasyEnsemble(3.37)	0.784	0.228	Not rejected for BPSO-Adaboost-KNN
PUSBE(3.05)	1.444	0.081	Not rejected for BPSO-Adaboost-KNN
CS-MCS(2.47)	2.455	0.08	Rejected for BPSO-Adaboost-KNN
ClusterBal(3.13)	2.319	0.032	Rejected for BPSO-Adaboost-KNN

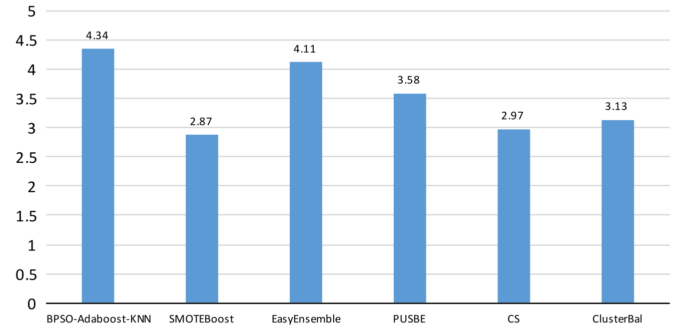

Fig. 7. Average rankings of BPSO-Adaboost-KNN and the state-of-the-arts.

Table 13
Oilsk81 well logging explanation results.

Reservoir number	AC (μs/m)	CNL (%)	RT (Ω. m)	POR (%)	SO (%)	PERM (mμm^2)	Conclusion
1	195	7.5	13.0	6.0	0	0	Dry layer
2	225	10.0	7.3	11.0	0	0	Water layer
3	230	14.0	5.5	12.0	0	0	Water layer
4	220	9.0	25.0	9.0	56	1.3	Oil layer
5	225	30.0	30.0	9.0	58	2.3	Oil layer
6	210	7.0	26.0	6.0	0	0	Dry layer
–	–	–	–	–	–	–	–
30	201	6.0	16.0	7.0	40	0.4	Inferior oil layer
31	213	9.5	12.0	9.0	61	2	Oil layer

Table 14
The distribution of each well logging.

Well number	The distribution of each sample (dry layer, water layer, oil layer, inferior oil layer)	IR
Oilsk81	(14, 2, 12, 3)	1:7
Oilsk82	(28, 2, 7, 11)	1:14
Oilsk83	(28, 3, 12, 7)	1:9
Oilsk84	(32, 6, 5, 9)	1:6
Oilsk85	(44, 4, 6, 11)	1:11

Table 15
The optimal feature sets in 5 test data.

Training set	The optimal feature set
Oilsk82	AC, SO
Oilsk83	POR, PERM
Oilsk84	AC, POR, SO
Oilsk85	AC, SO
Oilsk82-Oilsk85	AC, POR, SO

about is the location of oil layer. In this regard, BPSO-Adaboost-KNN has significant superiority.

5. Conclusions

This paper discusses a novel ensemble algorithm named of BPSO-Adaboost-KNN, which is deigned to solve multiple class imbalanced data problems. In this model, we use BPSO to select key features of datasets so that the classifier can ignore more noise. Considering traditional classifiers gain a poor performance when facing imbalanced data, we generate the Adaboost-KNN classifier by using boosting-by-resample strategy. Another significant contribute of this paper lies in employing a novel measure AUCarea as the criteria for selecting optimal sub-feature set. This metric not only has no bias toward the majority class but also can help us find which two classes are confused to distinguish by analyzing the single AUC values.

After a detail description of BPSO-Adaboost-KNN is carried out, we have compared our algorithm with other state-of-the-art algorithms. The results show that the proposed algorithm is comparable with other algorithms and superior to other algorithms in some cases. In experimental studies we also demonstrated each component of our ensemble model has significant contribution.

Besides, this paper has practical contribution since we applied the proposed algorithm to oil reservoir recognition. The largest advantage of BPSO-Adaboost-KNN in this application lies in detecting oil layer efficiently through logging data. The results show that BPSO-Adaboost-KNN has significant superiority when distinguishing oil layer from other layers.

In the future, multiple strategies such as cost sensitive learning and sampling technologies will be taken into account to deal with multi-class imbalanced data classification problems. More specifically, for oil reservoir recognition, when oil layer is misclassified, we would rather that the classifier assign oil layer samples into inferior oil layer than assign oil layer samples into other layer. This is because inferior oil layers are suspected oil layers, which means the companies may not abandon them immediately until it is confirmed there is no oil exist. This strategy can be implemented by cost sensitive learning, but the cost for each layer remains to be analyzed.

Table 16
Classification results of logging data in terms of accuracy and AUCarea.

Datasets	KNN		Adaboost-KNN		BPSO-KNN		BPSO-Adaboost-KNN	
	AUCarea	Accuracy	AUCarea	Accuracy	AUCarea	Accuracy	AUCarea	Accuracy
Oilsk82	0.5786	0.7708	0.8996	0.9117	0.9082	0.8750	**0.9743**	**0.9883**
Oilsk83	0.8639	0.9400	0.8791	0.9628	0.9082	0.9600	**1.0000**	**0.9976**
Oilsk84	0.8822	0.7500	0.9280	0.9615	0.9030	0.9231	**0.9853**	**0.9938**
Oilsk85	0.6629	0.7846	0.7311	0.9252	0.7779	0.9231	**0.9408**	**0.9882**
Oilsk82-Oilsk85	0.7844	0.8093	0.8005	0.9377	0.8944	0.8977	**0.9470**	**0.9839**

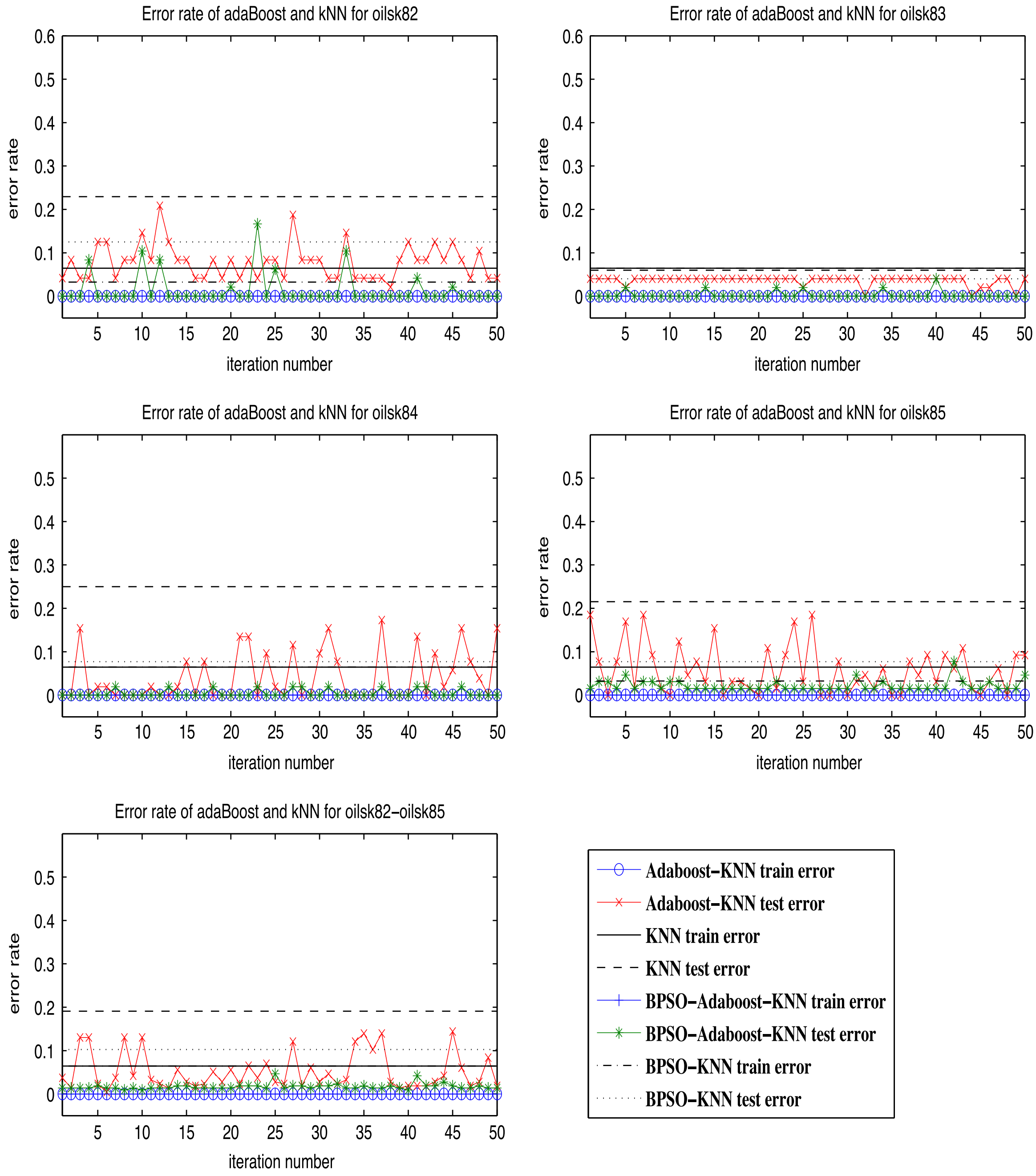

Fig. 8. Error rates of 50 times experiments.

Table 17
AUC of each pair of layers.

	Oilsk82		Oilsk83		Oilsk84		Oilsk85		Oilsk82–85	
	Our algorithm	KNN	Our algorithm	KNN	Our algorithm	KNN	Our algorithm	KNN	Our algorithm	KNN
Dry vs. water layer	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9995
Dry vs. oil layer	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Dry vs. inferior oil layer	1.0000	0.8636	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9596
Water vs. oil layer	**1.0000**	0.8636	1.0000	1.0000	**1.0000**	0.9167	**1.0000**	0.9090	**1.0000**	0.8965
Water vs. inferior oil layer	1.0000	0.7987	1.0000	1.0000	1.0000	0.9815	0.9090	0.9090	0.9793	0.8491
Oil vs. inferior oil layer	**0.9610**	0.3571	**1.0000**	0.8571	**0.9778**	0.8333	**0.9090**	0.5000	**0.9189**	0.7522

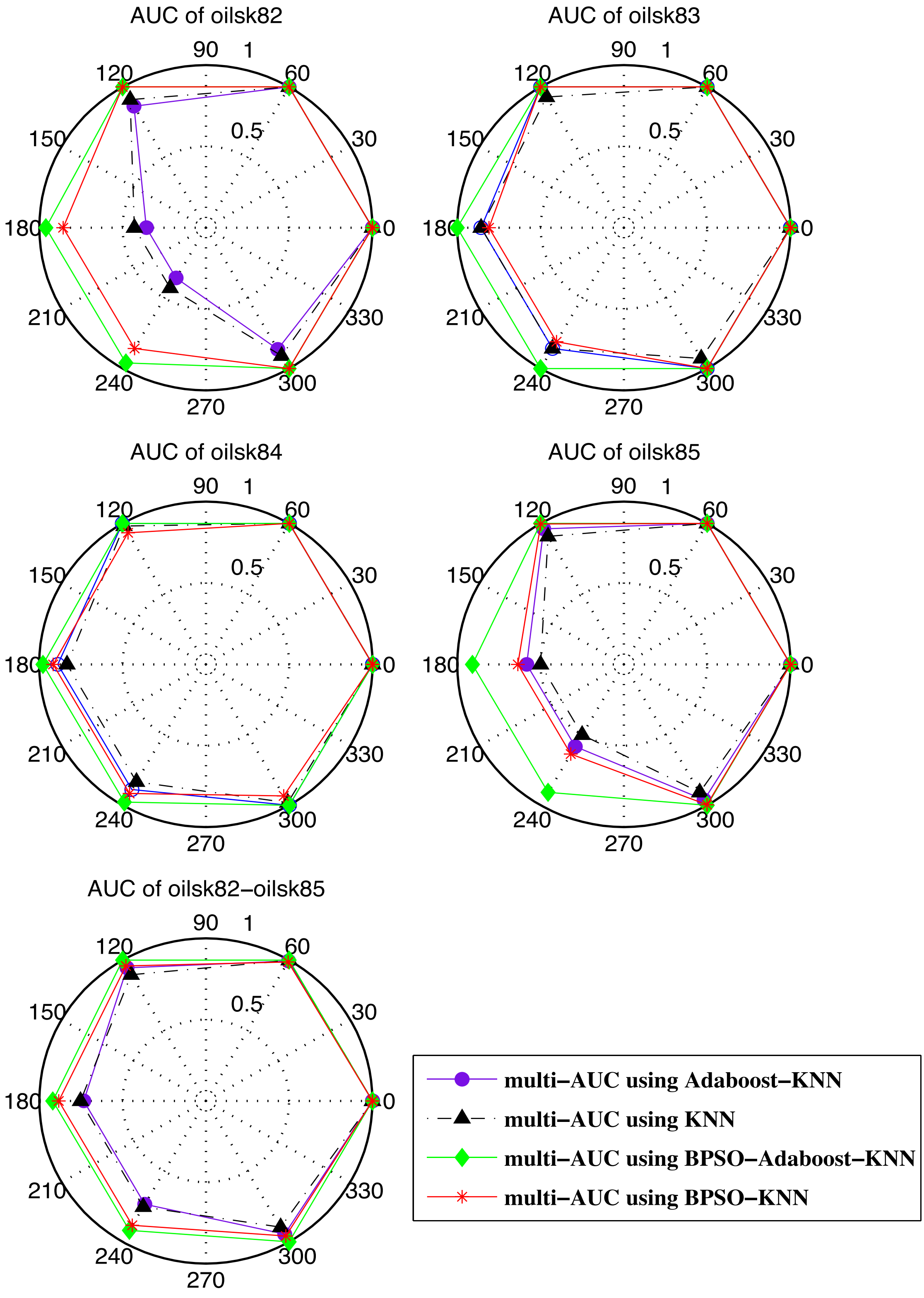

Fig. 9. AUC diagrams obtained in 5 test datasets.

Acknowledgments

This research has been supported by National Natural Science Fundation of China under Grant Nos. 71103163, 71573237; New Century Excellent Talents in University of China under Grant no. NCET-13-1012; Research Foundation of Humanities and Social Sciences of Ministry of Education of China No. 15YJA630019; Special Funding for Basic Scientific Research of Chinese Central University under Grant nos. CUG120111, CUG110411, G2012002A, CUG140604; Open Foundation for the Research Center of Resource Environment Economics in China University of Geosciences (Wuhan) (Grant no.: H2015004B); Structure and Oil Resources Key Laboratory Open Project of China under Grant no.TPR-2011-11.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.engappai.2015.09.011.

References

Alibeigi, M., Hashemi, S., Hamzeh, A., 2012. DBFS: An effective Density Based Feature Selection scheme for small sample size and high dimensional imbalanced data sets. Data Knowl. Eng., 67–103.

Alberto, F., Victoria, L., Mikel, G., et al., 2013. Analysing the classification of imbalanced data-sets with multiple classes: binarization techniques and ad-hoc approaches. Knowl.-Based Syst. 42, 97–110.

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

Bin, W., Qinke, P., Jing, Z., et al., 2012. A binary particle swarm optimization algorithm inspired by multi-level organizational learning behavior. Eur. J. Oper. Res. 219, 224–233.

Bache, K., Lichman, M., 2013. UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA.

Bland, J.M., Altman, D.G., 1986. Statistical methods for assessing agreement between two methods of clinical measurement. Lancet 8476, 307–310.

Chawla, N.V., Bowyer, K.W., Hall, L.O., 2002. SMOTE: synthetic minority oversampling technique. Artif. Intell. Res. 16, 321–357.

Cao, Miao, Liu, et al., 2013. Advance and prospects of AdaBoost algorithm. Acta Autom. Sin. 39 (6), 745–758.

Chawla, N.V., Japkowicz, N., Kotcz, A., 2004. Editorial: Special Issue on Learning from Imbalanced Data Sets. ACM SIGKDD Explor. Newsl. 6 (1), 1–6.

David, J.H., Robert, J.T., 2001. A Simple generalization of the area under the ROC curve for multiple class classification problems. Mach. Learn. 45, 171–186.

Díez-Pastor, J.F., Rodríguez, J.J., García-Osorio, C., Kuncheva, L.I., 2015. Random balance: ensembles of variable priors classifiers for imbalanced data. Knowledge-Based Systems 85, 96–111.

Searle, S.R., 1987. Linear Models for Unbalanced Data. Wiley.

Feng, G.Q., Li, Y., Tan, D.H., 1999. Application of fuzzy closeness degree in reservoir recognition. J. Southwest Petrol. Ins. 21 (4), 46–49.

Freund, Yoav, 1995. Boosting a weak learning algorithm by majority. Inf. Comput. 121 (2), 256–285.

Freund, Y., Schapire, R.E., 1996. Experiments with a New boosting algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning, pp. 148-156.

Fawcett, Tom, 2006. An introduction to ROC analysis. Pattern Recognit. Lett. 27, 861–874.

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

Ferri, C., Hernández-orallo, J., Salido, M.A., 2003. Volume under the ROC Surface for Multi-class Problems. Exact computation and evaluation of approximations. In: Proceedings of 14th European Conference on Machine Learning, pp. 108–120.

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

Guo, H.X., Liao, X.W., Zhu, K.J., et al., 2011. Optimizing reservoir features in oil exploration management based on fusion of soft computing. Appl. Soft Comput. 11, 1144–1155.

Galar, M., Ferna ́ndez, A., Barrenechea, E., 2012. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans. Syst. Man Cybern C: Appl. Rev. 42 (4), 463–484.

Gao, Ming, Hong, Xia, Chris, J., 2014. Construcion of neurofuzzy models for imbalanced data classification. IEEE Trans. Fuzzy Syst. 22 (6), 1473–1488.

Md Hassan, Kotagiri, R., Chandan, K. et.al., June, 21–24, 2010. A novel scalable multi-class ROC for effective visualization and computation. Advances in Knowledge Discovery and Data Mining. 14th Pacific-Asia Conference, .

He, H., Bai, Y. et al., 2008. Adasyn: adaptive synthetic sampling approach for imbalanced learning. In: Proceedings of IEEE International Joint Conference on Neural Networks, pp. 1322–1328.

Holm, S., 1979. A simple sequentially rejective multiple test procedure, Scand. J. Stat 6, 65–70.

Inbarani, H., Azar, A., Jothi, G., 2014. Supervised hybrid feature selection based on PSO and rough sets for medical diagnosis. Comput. Methods Progr. Biomed. 113, 175–185.

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

Kennedy, J., Eberhart, R.,1995. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948.

Kennedy, J., Eberhart, R.C., 1997. A discrete binary version of the particle swarm algorithm. In: Proceedings of IEEE International Conference on Systems Man and Cybernetics Computational Cybernetics and Simulation.

Krawczyk, B., Schaefer, G., 2013. An improved ensemble approach for imbalanced classification problems, In: Proceedings of 8th IEEE International Symposium on Applied Computational Intelligence and Informatics.

Krawczyk, B., Wozniak, M., Herrera, F., 2014. Weighted one-class classification for different types of minority class examples in imbalanced data, Computational Intelligence and Data Mining (CIDM). IEEE Symposium on, 337–344.

Krawczyk, B., Woźniak, M., 2015. Hypertension type classification using hierarchical ensemble of one-class classifiers for imbalanced data. Advances in Intelligent Systems Computing.

López, V., Fernández, A., del Jesuse, M.J., et al., 2013. A hierarchical genetic fuzzy system based on genetic programming for addressing classification with highly imbalanced and borderline data-sets. Knowl.-Based Syst. 38, 85–104.

López, V., Fernández, A., García, S., et al., 2013. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. Inf. Sci. 250, 113–141.

López, V., Río, S., Benítez, J.M., et al., 2015. Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data. Fuzzy Sets and Syst 258, 5–38.

Liu, X.Y., Wu, J., Zhou, Z.H., 2009. Exploratory undersampling for class-imbalance learning. IEEE Trans. Syst. Man Cybern. Part B: Cybern. 39 (2), 539–550.

López, V., Fernández, A., García, S., et al., 2006. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. Inf. Sci. 250 (2013), 113–141.

Lachiche, N., Flach, P.A., 2003. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. ICML-2003.

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

Maldonado, S., Weber, R., Famili, F., 2014. Feature selection for high-dimensional class-imbalanced data sets using support vector machines. Inf. Sci. 286, 228–246.

Martino, M.D., Fernández, A., Iturralde, P., et al., 2013. Novel classifier scheme for imbalanced problems. Pattern Recognit. Lett. 34, 1146–1151.

Nitesh, V.C., Aleksandar, L., Lawrence, O.H., et.al., 2003. SMOTEBoost: improving prediction of the minority class in boosting. In: Proceedings of 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 107–119.

Naimul, M.K., Riadh, K., Imran, S.A., 2014. Covariance-guided One-Class Support Vector Machine. Pattern Recognit. 47 (6), 2165–2177.

Nakas, C.T., Yiannoutsos, C.T., 2004. Ordered multiple-class ROC analysis with continuous measurements. Stat. Med. 23 (22), 3437–3449.

Nanni, L., Fantozzi, C., Lazzarini, N., 2015. Coupling different methods for overcoming the class imbalance problem. Neurocomputing 158, 48–61.

Prati, R.C., Gustavo, E., Silva, Diego F., 2015. Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. Knowl. Inf. Syst. . http://dx.doi.org/10.1007/s10115-014-0794-3

Richard, M., Jonathan, E., 2006. Multi-class ROC analysis from a multi-objective optimization perspective. Pattern Recognit. Lett. 27, 916–927.

Sun, Z., Song, Q., Zhu, X., et al., 2015. A novel ensemble method for classifying imbalanced data. Pattern Recognit. 48, 1623–1637.

Sheskin, D., 2006. Handbook of Parametric and Nonparametric Statistical Procedures. Chapman & Hall/CRC, London, U.K..

Schapire, Robert E., Singer, Yoram, 1999. Improved boosting algorithms using confidence-rated pre-dictions. Mach. Learn. 37 (3), 297–336.

Shanab, A., Khoshgoftaar, T., Wald, R. et al., 2011. Comparison of approaches to alleviate problems with high-dimensional and class-imbalanced data. In: Proceedings of IEEE International Conference on Information Reuse and Integration, pp. 234–239.

Sathiya, K., Olivier, C., Dennis, D., 2006. Building support vector machines with reduced classifier complexity. J. Mach. Learn. Res. 7, 1493–1515.

Tian, J., Gu, H., Liu, W., 2011. Imbalanced classification using support vector machine ensemble. Neural Comput. Appl. 20 (2), 203–209.

Thai-Nghe, N., Gantner, Z. Schmidt-Thieme, L., 2010. Cost-sensitive learning methods for imbalanced data. In: Proceedings of IEEE International Joint Conference Neural Networks (IJCNN), pp. 1–8.

Thomas, L., Robert, P.W.D., 2006. A simplified extension of the Area under the ROC to the multiclass domain. In: Proceedings of 17th Annual Symposium of the Pattern Recognition Association of South Africa. PRASA, pp. 241–245.

Vorraboot, P., Rasmequan, S., Chinnasarn, K., et al., 2015. Improving classification rate constrained to imbalanced data between overlapped and non-overlapped regions by hybrid algorithms. Neurocomputing 152, 429–443.

Wang, X., Matwin, S., Japkowicz, N., Liu, X., 2013. Cost-sensitive boosting algorithms for imbalanced multi-instance datasets. In: Proceedings of the Canadian Conference on Artificial Intelligence, pp. 174–186.

Wilcoxon, F., 1945. Individual comparisons by ranking methods. Biom. Bull. 6, 80–83.

Yin, L., Ge, Y., Xiao, K., et al., 2013. Feature selection for high-dimensional imbalanced data. Neurocomputing 105, 3–11.

Yin, L., Ge, Y., Xiao, K., 2013. Feature selection for high-dimensional imbalanced data. Neurocomputing 105 (3), 3–11.