Neural Comput & Applic
DOI 10.1007/s00521-016-2342-4

ORIGINAL ARTICLE

Ensemble weighted extreme learning machine for imbalanced data classification based on differential evolution

Yong Zhang[1,2] · Bo Liu[1] · Jing Cai[1] · Suhua Zhang[1]

Received: 15 January 2016 / Accepted: 9 May 2016

Abstract Extreme learning machine for single-hidden-layer feedforward neural networks has been extensively applied in imbalanced data learning due to its fast learning capability. Ensemble approach can effectively improve the classification performance by combining several weak learners according to a certain rule. In this paper, a novel ensemble approach on weighted extreme learning machine for imbalanced data classification problem is proposed. The weight of each base learner in the ensemble is optimized by differential evolution algorithm. Experimental results on 12 datasets show that the proposed method could achieve more classification performance compared with the simple vote-based ensemble method and non-ensemble method.

Keywords Extreme learning machine · Differential evolution · Ensemble · Class imbalanced learning

1 Introduction

Imbalanced data can be found in many real fields such as bioinformatics, fraud detection, oil spill detection and image annotation [1, 2]. In recent years, the classification of imbalanced data has become one of the key issues in machine learning field and attracted tremendous attention. Simply speaking, the class imbalance learning (CIL) problem occurs when the size of one class is larger than those of the other classes.

Consider a binary classification problem with a large amount of samples labeled as minority class and very few samples labeled as majority class. We are usually more concerned with the minority class, because it contains more important information than the majority class. Therefore, we are also more concern about the classification performance of the minority class. However, most standard learning algorithms, such as decision trees, *k*-nearest neighbors, neural networks and support vector machine, tend to be biased toward to the majority class and ignore the minority class when dealing with imbalanced datasets [3, 4]. The reason is that these algorithms are designed originally based on the assumption that the size of each class is relatively balanced and misclassification costs are equal within the whole datasets [3].

Many approaches have been developed to handle the CIL problem. Among these approaches, re-sampling is one of the widely used methods to copy with the CIL problem, which can reduce the class imbalance ratio to some extent [5]. Re-sampling includes under-sampling and oversampling. Cost-sensitive learning is also used to deal with the CIL problem by punishing different costs for the two classes [6]. Generally, high cost is assigned to the minority class, while low cost is assigned to the majority class. In addition, ensemble learning is another commonly employed approach to deal with the CIL problem [7]. Ensemble learning usually needs to train multiple learners, called base learners, and assemble them to solve the classification problem. It is well known that a classification model assembling several learners may decrease the chances of overfitting and help to improve generalization performance.

Many kinds of ensemble learning approaches have been developed, such as bagging, boosting and stacking [8],

✉ Yong Zhang
zhyong@lnnu.edu.cn

[1] School of Computer and Information Technology, Liaoning Normal University, No. 1, Liushu South Street, Ganjingzi District, Dalian 116081, Liaoning Province, China

[2] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China

Published online: 18 May 2016

which combine several base learners according to different strategies. Simple voting and weighted voting are two widely used strategies in the combination of base learners. Simple voting strategy decides the final category of a new sample based on the predicted classes with the most votes, while weighted voting strategy needs the weights assigning to each base learner and votes by weighting to decide the category of a new sample. Obviously, the weights should vary among the different output classes in each base learner. Therefore, it is still an important and meaningful problem to determine the weights of base learners to achieve better classification performance. Selection of the weights can be taken as a parameter optimization problem, which can usually be solved by particle swarm optimization (PSO) and genetic algorithm (GA) [9, 10]. Moreover, differential evolution (DE) is a very simple population-based evolutionary method to optimize a numerical problem [11] and has been successfully used in many fields, such as parallel computing, constrained optimization and multi-objective optimization [12]. This paper tries to use DE method to optimize the weights of base learners in the ensemble.

In recent years, extreme learning machine (ELM), proposed by Huang et al. [13], is a simply and efficient learning method for single-hidden-layer feedforward neural networks (SLFNs) and has been used to deal with the CIL problem [14]. However, a single ELM classifier is lack of stability to some extent when solving the class imbalance problem. Therefore, traditional ELM method has also employed ensemble strategy to carry out the research [15, 16]. Inspired by the boosting weighted ELM (WELM) for imbalanced learning [15], the ensemble strategy could be further improved by taking into account the different activation function of ELM. In the ensemble learning, base learners should be as diverse as possible. This paper presents a novel DE-based ensemble learning strategy to handle the classification problem of imbalance data. In our strategy, in order to embody the diversity, we select several WELMs with different activation functions as base learners. Moreover, in the ensemble learning process, DE algorithm, as a widely used optimization method, is used to optimize the weight of each base learner. Experimental results on 12 binary datasets from KEEL repository indicate that under the geometric mean (*G-mean*) metric, the proposed method could achieve better classification performance than other compared methods.

This paper is organized as follows. In the next section, we discuss the related work. Section 3 introduces some preliminaries of ELM and DE. The DE-based WELM ensemble algorithm is presented in Sect. 4. Section 5 reports our experimental results on 12 datasets and discusses the effectiveness of the proposed algorithm. In the last section, we conclude the paper with some remarks.

2 Related work

A variety of ELM-based approaches has been developed for handling the CIL problem. Zong et al. [14] proposed the WELM method to deal with the imbalanced data classification. WELM can assign different weights for each training sample based on two different strategies. A weighted online sequential ELM (WOS-ELM) was also presented for the CIL problem [17], which can learn samples in chunk by chunk (a block of data) or one by one. On the basis of WELM, semi-supervised learning issue is further investigated to handle the CIL problem [18]. Xia et al. [19] presented a kernel clustering-based possibilistic fuzzy ELM algorithm to deal with the CIL problem. The results show that the proposed method is very effective for learning imbalanced data. Mao et al. [20] proposed an ELM-based model with sparse-weighting strategy for sequential data imbalance problem. The weight of each sequential sample is dynamically assigned according to the change of sensitivity and specificity.

Ensemble approaches are also extensively used in the imbalance data classification. Compared to approaches with a single classifier, ensemble approaches tend to better deal with the CIL problem. Inspired by WOS-ELM, Mirza et al. [21] proposed an ensemble of subset online sequential ELM (ESOS-ELM) to handle CIL from a concept-drifting data stream. In the proposed method, a sample belonging to the minority class is learned by m classifiers, while a sample belonging to the majority class is learned by a single classifier, where m is the imbalance ratio. Cao et al. [22] presented the voting-based ELM ensemble learning. Li et al. [15] embedded WELM seamlessly into an improved AdaBoost model and proposed a boosting weighted ELM, to solve the imbalanced data classification problem.

Moreover, several optimization strategies have been widely used in parameter selection, such as GA, PSO and DE. Zhu et al. [10] presented a hybrid learning algorithm named E-ELM, which employs DE algorithm and Moore–Penrose generalized inverse to process the input weights and the output weights. The E-ELM can achieve higher generalization performance than several comparison methods. Feng et al. [23] presented an ES-ELM method which uses the crossover mechanism derived from the genetic algorithm to select the optimal hidden nodes for ELM. Cao et al. [24] proposed a self-adaptive evolutionary ELM (SaE-ELM) for SLFNs, in which the number of hidden nodes is optimized by the self-adaptive DE method.

However, these strategies are mainly used to parameter optimization of ELM and seldom optimize the weights of base learners in the ensemble.

3 Preliminaries

In this section, we first give a brief description of ELM and WELM. And then, we briefly review DE algorithm.

3.1 ELM

The SLFNs are extensively used for many regression and classification problems due to their universal approximation ability [25]. Aiming at the training of SLFNs, gradient-based learning methods are widely applied, such as backpropagation (BP) and Levenberg–Marquardt (LM). However, the above gradient-based methods have the disadvantages of relatively slow convergence speed and are apt to be stuck in the local minimum. To overcome these shortcomings, Huang et al. [13] presented ELM algorithm for SLFNs. The ELM can randomly choose the input weights and hidden biases which do not need to be adjusted during the training process. The output weights can be analytically determined with Moore–Penrose generalized inverse of the hidden-layer output matrix.

Consider a classification problem with N different samples $(\mathbf{x}_i, \mathbf{t}_i)_{i=1}^{N}$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^{\mathrm{T}}$ is the ith sample with n-dimensional features and $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^{\mathrm{T}}$ indicates the actual output values of $\mathbf{x}_i$. The output of a standard SLFN with L hidden nodes and activation function $g(\cdot)$ is

$$o_i = \sum_{j=1}^{L} \beta_j g(a_j, b_j, x_i), \quad i = 1, 2, \ldots, N \tag{1}$$

where $a_j = [a_{j1}, a_{j2}, \ldots, a_{jn}]^{\mathrm{T}}$ is the weight vector connecting the jth hidden node with the input nodes, $\beta_j = [\beta_{j1}, \beta_{j2}, \ldots, \beta_{jm}]^{\mathrm{T}}$ is the output weight connecting the jth hidden node with the output nodes, b_j is the bias of the jth hidden node and o_i is the ith output node.

ELM states that an SLFN with L hidden nodes can approximate these N samples with zero error means that $\sum_{i=1}^{N} \|o_i - t_i\| = 0$ [24]. In other words, there exist β_j, a_j and b_j such that

$$\sum_{j=1}^{L} \beta_j g(a_j, b_j, x_i) = t_i, \quad i = 1, 2, \ldots, N \tag{2}$$

Equation (2) can be rewritten in a more compact form as

$$\mathbf{H}\beta = \mathbf{T} \tag{3}$$

where $\mathbf{H} = \{h_{ij}\}$ is the hidden-layer output matrix of the network and $h_{ij} = g(a_j, b_j, \mathbf{x}_i)$, $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_L]^{\mathrm{T}}$ is the output weight matrix and $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N]^{\mathrm{T}}$ is the target matrix of the output layer.

Thus, Eq. (3) becomes a linear system and the output weight can be analytically determined by a least-square solution as follows.

$$\hat{\beta} = \mathbf{H}^{+}\mathbf{T} \tag{4}$$

where $\mathbf{H}^{+}$ is the Moore–Penrose generalized inverse of $\mathbf{H}$. The obtained solution in Eq. (4) indicates a good generalization performance of ELM on classification and regression tasks [26].

Activation function $g(\mathbf{x})$ in the matrix $\mathbf{H}$ actually maps the sample $\mathbf{x}$ from the original data space to the hidden-layer space. The hidden node parameters (a, b) in $g(\mathbf{x})$ are randomly generated according to any continuous probability distribution. If a function satisfies ELM universal approximation capability theorems [27], it can be one of the activation functions. Table 1 lists several commonly used activation functions in ELM algorithm [28].

3.2 Weighted ELM for imbalanced data classification

For imbalanced data classification problem, Zong et al. [14] presented a unified solution of WELM. Each sample is assigned a specific weight to indicate its effect on the classification results. An optimization problem for WELM is summarized as follows, which can maximize the marginal distance as well as minimize the weighted cumulative error.

$$\min_{\beta} \frac{1}{2}\|\beta\|^2 + C\mathbf{W}\frac{1}{2}\sum_{i=1}^{N}\|\xi_i\|^2 \text{ subject to } h(\mathbf{x}_i)\beta = t_i^T - \xi_i^T, i = 1, \ldots, N \tag{5}$$

where C is a trade-off constant between maximizing the marginal distance and minimizing the cumulative error. $\mathbf{W}$ is an $N \times N$ diagonal matrix associated with each training sample. For the CIL problem, WELM usually assigns a larger value of W_{ii} to sample $\mathbf{x}_i$ belonging to minority class and assigns a smaller value of W_{ii} to sample $\mathbf{x}_i$ belonging to majority class. Therefore, the impact of minority class is strengthened, while the impact of majority class is weakened.

For formula (5), we can obtain its equivalent dual optimization form according to KKT theorem and the corresponding solution is represented as

$$\hat{\beta} = \mathbf{H}^{+}\mathbf{T} = \begin{cases} \mathbf{H}^{\mathrm{T}}\left(\frac{I}{C} + W\mathbf{H}^{\mathrm{T}}\right)^{-1}\mathbf{W}\mathbf{T}, & \text{when } N < L \\ \left(\frac{I}{C} + \mathbf{H}^{\mathrm{T}}W H\right)^{-1}\mathbf{W}\mathbf{H}^{\mathrm{T}}\mathbf{T}, & \text{when } N \geq L \end{cases} \tag{6}$$

Table 1 The commonly used activation functions in ELM

Gaussian function	$g(a,b,\mathbf{x}) = \exp(-b\|\mathbf{x}-a\|)$
Sigmoid function	$g(a,b,\mathbf{x}) = \frac{1}{1+\exp(-(a\times\mathbf{x}+b))}$
Hyperbolic tangent function	$g(a,b,\mathbf{x}) = \frac{1-\exp(-(a\cdot\mathbf{x}+b))}{1+\exp(-(a\cdot\mathbf{x}+b))}$
Multi-quadric function	$g(a,b,x) = (\|x-a\|+b^2)^{1/2}$
Hard limit function	$g(a,b,\mathbf{x}) = \begin{cases} 1, & \text{if } a\times\mathbf{x}+b\le 0 \\ 0, & \text{otherwise} \end{cases}$
Triangular basis function	$g(a,b,\mathbf{x}) = \begin{cases} 1-\|a\cdot\mathbf{x}+b\|, & \text{if } \|a\cdot\mathbf{x}+b\|\le 1 \\ 0, & \text{otherwise} \end{cases}$
Cosine function	$g(a,b,\mathbf{x}) = \cos(a\times\mathbf{x}+b)$
Sine function	$g(a,b,\mathbf{x}) = \sin(a\times\mathbf{x}+b)$

There are multiple approaches to computing weights $\mathbf{W}$ in Eq. (6). In this paper, we assign an equivalent weight to each sample in the same class according to the number of samples in the class [14]. That is to say, we set $W_{ii} = 1/\#(t_i)$ in Eq. (6), where $\#(t_i)$ denotes the number of samples from class t_i.

3.3 Differential evolution

DE algorithm is an efficient and robust evolutionary algorithm developed by Storn and Price [11]. It has been widely used in tuning the parameters of neural networks. Three are three parameters in DE algorithm, including the population size NP, mutation scaling factor F and crossover rate CR. DE algorithm starts with a randomly generated initial population of N individuals (candidate solutions) $X_{i,G}$, $i = 1, \ldots, NP$, where the index i indicates the ith candidate solution of the population at generation G. Each individual is defined as a D-dimensional vector $X_{i,G} = [x_{i,G}(1), x_{i,G}(2), \ldots, x_{i,G}(j), \ldots, x_{i,G}(D)]$. And then, these individuals are gradually improved through three key operations of DE, namely mutation, crossover and selection. The basic learning process of DE involves the iteration of these three operations:

Mutation For each target vector $X_{i,G}$, $i = 1, \ldots, NP$, a mutant vector $V_{i,G}$ is generated according to the most common DE version [29]

$$\mathrm{DE/rand/1}: V_{i,G} = X_{r_1,G} + F \times \left(X_{r_2,G} - X_{r_3,G}\right) \tag{7}$$

with random and mutually different indices $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$. In fact, mutant vector $V_{i,G}$ is a linear combination of three randomly chosen vectors. In Eq. (7), F is a scaling factor parameter and its value is within the range [0, 2], typically less than 1.

Crossover In this operation, the D-dimensional trial vector $T_{i,G} = [t_{i,G}(1), t_{i,G}(2), \ldots, t_{i,G}(j), \ldots, t_{i,G}(D)]$ is formed in accordance with the target vector $X_{i,G}$ and its corresponding mutant vector $V_{i,G}$ as follows:

$$t_{i,G}(j) = \begin{cases} v_{i,G}(j), & \text{if } (\mathrm{rand}_{i,j}[0,1] \le CR \quad \text{or} \quad j = j_{\mathrm{rand}}) \\ x_{i,G}(j), & \text{otherwise} \end{cases} \tag{8}$$

where $CR \in [0, 1]$ is a crossover control parameter, called crossover rate. The value of CR stands for the probability that a component of the trial vector is chosen from a mutant vector. $\mathrm{rand}_{i,j} \in [0, 1]$ is a uniformly distributed random number, for each jth component of the ith vector. $j_{\mathrm{rand}} \in [1, 2, \ldots, D]$ is a randomly chosen index to make sure that at least one component in $T_{i,G}$ is different from the components of $V_{i,G}$.

Selection After the mutation and crossover operations, the DE algorithm will make a comparison between the trial vector $T_{i,G}$ and its target vector $X_{i,G}$. If the trial vector obtains a better objective function value than the target vector, then it replaces the target vector surviving to the next generation $(G + 1)$. Otherwise, the target vector is retained in the next generation $(G + 1)$. The selection operation is described as:

$$X_{i,G+1} = \begin{cases} T_{i,G}, & \text{if } f(T_{i,G}) \le f(X_{i,G}) \\ X_{i,G}, & \text{otherwise} \end{cases} \tag{9}$$

where $f(.)$ is the objective function to be optimized and ensures a member of the next generation is the fittest individual. From Eq. (9), we can see the best population member is always preserved. If the trial vector $T_{i,G}$ is better than the target vector $X_{i,G}$, namely $f(T_{i,G}) \le f(X_{i,G})$, then it replaces the target vector in the next generation.

After the iteration process, the best individual in the population provides the lowest objective function value and is selected as the weights of base learners in this paper. Therefore, the best classification performance can be obtained for the ensemble learning.

4 The DE-based WELM ensemble learning algorithm

In this section, we first analyze the ensemble method of several WELMs and then give a detailed description of DE-based WELM ensemble learning algorithm.

4.1 Ensemble of WELMs

It is stated that an ensemble learning model can be better than a single learning model for solving the CIL problem [15]. There are two approaches to assigning the weights of base learners. One is to assign an equal weight to each base learner, and the other is to assign different weights for base learners. In our proposed ensemble learning strategy, we select several WELMs with different activation functions as the base learners. Their weights α_k ($k = 1, \ldots, K$, where K is the number of base learners) in the ensemble form a vector $\Lambda = (\alpha_1, \alpha_2, \ldots, \alpha_j, \ldots, \alpha_K)$, which is taken as an individual of a population in DE algorithm and to be optimized. All α_k in the individual of the population are randomly initialized within the range of [0, 1].

Suppose that there are l categories and K base learners. A new sample $\mathbf{x}^{\text{new}}$ will be classified according to

$$c_{\text{new}} = \arg \max_{i \in [1,2,\ldots,l]} \sum_{k=1}^{K} \alpha_k [\Omega_k(\mathbf{x}^{\text{new}}) = i] \tag{10}$$

where Ω_k represents a base learner. If sample $\mathbf{x}^{\text{new}}$ is classified into the class i by base learner Ω_k, the value of $c_{\text{new}} = \arg \max_{i \in [1,2,\ldots l]} \sum_{k=1}^{k} \alpha_k [\Omega_k(\mathbf{x}^{new}) = i]$ is 1. Otherwise, its value is 0.

4.2 DE-based WELM ensemble learning algorithm

In the initial phase, we divide the whole dataset into three subsets, including a training set, a validation set and a test set. In the optimization process, to alleviate possible overfitting, a validation set is used instead of the whole training dataset.

Each individual consists of the weights of base learners and is generated randomly in the first generation. These individuals form an initial population. At each iteration, the population evolves through mutation and crossover, and then, better individuals are selected based on the fitness of each individual. The fitness is evaluated as the *G-mean* in this paper. *G-mean* will be introduced in Sect. 5.1.

After the optimization phase, a population of candidate weight vectors is generated and we select an individual with the best fitness value as weights of base learners in the ensemble. The procedure of DE-based WELM ensemble learning algorithm is shown in Algorithm 1 as follows.

Algorithm 1: The DE-based WELM ensemble learning algorithm

Input: A dataset $\{\mathbf{x}_i, \mathbf{t}_i\}_{i=1}^{N}$; number of base learners: K; number of the maximum iterations: G_{max}; a series of DE parameters: mutation factor F, crossover rate CR, and population size NP.

Output: The prediction of the testing set.

**** (1) Partitioning dataset and training phase ****

Split the dataset into three subsets: a training set $\{\mathbf{x}_i, \mathbf{t}_i\}_{i=1}^{N^{tr}}$, a validation set $\{\mathbf{x}_i, \mathbf{t}_i\}_{i=1}^{N^{va}}$, and a test set $\{\mathbf{x}_i, \mathbf{t}_i\}_{i=1}^{N^{test}}$, where N^{tr}, N^{va} and N^{test} represent the numbers of the training set, the validation set and the test set respectively, and $N^{tr}+N^{va}+N^{test}=N$.

for (k=1; k<=K; k++) **do**

Train the kth WELM base learner Ω_k with different activation function g($\mathbf{x}$) independently using the training set $\{\mathbf{x}_i, \mathbf{t}_i\}_{i=1}^{N^{tr}}$;

end for

**** (2) Weighting optimization phase based on DE ****

Initialization(); {Generate uniformly distributed random population of N individuals X_G= {$X_{1,G}$, $X_{2,G}$, …, $X_{i,G}$ …, $X_{N,G}$}, where $X_{i,G}$=[$x_{i,G}$(1), $x_{i,G}$(2), ..., $x_{i,G}$(j),…, $x_{i,G}$(K)] is a vector representing the weights $\Lambda = (\alpha_1, \alpha_2, ..., \alpha_j, ..., \alpha_K)$ of K base learners.}

Set the generation iterator G=0;

while G<G_{max} **do**

for (i=0; i<NP; i++) **do**

Select random indexes r1, r2, and r3 to be different from each other and from the index i;

Compute a mutant vector $V_{i,G}$ using Eq. (7);

Generate random number j_{rand};

for (j = 0; j < K; j++) **do**

Decide trial individual $T_{i,G}$ using Eq. (8);

end for

for each validation sample $\mathbf{x}^{va}$ **do**

Predict K labels of $\mathbf{x}^{va}$ using K base learners Ω_k, k=1, …, K;

Consider $X_{i,G}$ and $U_{i,G}$ as the weights Λ of K base learners respectively, and output the corresponding weighted voting label of $\mathbf{x}^{va}$: $\hat{t}^{va} = \arg\max_i \sum_{k=1}^{K} \alpha_k [\Omega_k(x^{va}) = i]$, i=1, …, l, where l is the category number of dataset;

end for

Compute *G-mean*s on the validation set as the fitness values $f(U_{i,G})$ and $f(X_{i,G})$, and update the vector $X_{i,G+1}$ of the next generation (G+1) using Eq. (9);

end for

Update generation iterator G=G+1;

end while

$\Lambda^{best} = X_{i,G}$ with the best fitness value;

**** (3) Test phase ****

for each testing sample $\mathbf{x}^{test}$ **do**

Predict K labels of $\mathbf{x}^{test}$ using K base learners Ω_k, k=1, …, K;

Output the weight voted label of $\mathbf{x}^{test}$: $\hat{t}^{test} = \arg\max_{i \in [1,2,...,l]} \sum_{k=1}^{K} \alpha_k^{best} [\Omega_k(\mathbf{x}^{test}) = i]$, where l is the category number of data set and $\Lambda^{best} = (\alpha_1^{best}, \alpha_2^{best}, ..., \alpha_j^{best}, ..., \alpha_K^{best})$;

end for

5 Experiments and analysis

5.1 Performance metrics for imbalance data classification

Accuracy is an important metric for evaluating the model performance in the classification problem. However, it is not a good metric for the imbalanced data classification problem. To make a fair comparison of performance, several measures have been developed to handle the CIL problem, including true-positive rate, *G-mean*, *F-measure* and the receiver operating characteristic curve (*ROC*). To better interpret these metrics for a binary classification problem, we give a confusion matrix to describe the result of classification which is shown in Table 2. There are four categories in the result, called *TP* (true positive), *FN* (false negative), *FP* (false positive) and *TN* (true negative).

This paper uses *G-mean* instead of accuracy to evaluate the classification performance. *G-mean* is defined by two parameter called sensitivity and specificity. Sensitivity shows the performance of the positive class, which is also called true-positive rate or the recall rate *R*. It is defined as follows:

$$P_{\text{sen}} = R = \text{TP}/(\text{TP} + \text{FN}). \tag{11}$$

Specificity, or true-negative rate, shows the performance of the negative class, which is defined as

$$P_{\text{spe}} = \text{TN}/(\text{TN} + \text{FP}). \tag{12}$$

For binary classification problem, *G-mean* is defined as the square root of ($P_{\text{sen}} \times P_{\text{spe}}$) as follows:

$$G - mean = \sqrt{P_{\text{sen}} \times P_{\text{spe}}} \tag{13}$$

Obviously, *G-mean* would be as low as 0 when the accuracy for the minority class is 0. Therefore, *G-mean* balances accuracies between the two classes and makes a more fair comparison of positive class and negative class.

5.2 Datasets

To compare the proposed DE-based WELM ensemble learning algorithm with other learning algorithms, 12 datasets from KEEL dataset repository [30] are used for binary classification problem, which are selected for their heterogeneity according to sizes, number of attributes and imbalance ratio. Table 3 shows the specification of 12 datasets listed in ascending order by the imbalance ratio.

Table 2 Confusion matrix

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

5.3 Experimental results and analysis

To show the efficiency of the proposed algorithm, we compare our DE-based WELM ensemble learning algorithm (DE-based ensemble) with several related algorithms, including vote-based WELM ensemble learning algorithm (vote-based ensemble) and non-ensemble WELM learning algorithm (non-ensemble). Vote-based ensemble is similar to the DE-based ensemble. It uses a simple vote strategy to decide the final category of each sample. Each base learner assigns the same weight in vote-based ensemble, while the different weights are assigned to base learners in our proposed DE-based ensemble. Non-ensemble does not use ensemble strategy, and only uses WELM to average the classification results.

In our proposed ensemble approach, we select 5 functions from Table 1 as activation functions, such as Gaussian function, sigmoid function, sine function, hard limit function and triangular basis function. Three methods in comparison use the same way to process all datasets. For each dataset, we use fivefold cross-validation and the result is averaged over 5 runs.

The number of base learners is set to be 20 in both DE-based ensemble and vote-based ensemble. For non-ensemble, we also record the average result of 20 base learners in each run. In DE optimization phase of DE-based ensemble, the dimension of the population is 20 corresponding to 20 base learners. DE optimization phase will be terminated when it reaches the maximum iteration number, which is set to be 200 in our experiments. The population size is set to be 100. The cross-rate *CR* is set to be 0.8.

Table 3 Dataset summary

Dataset	#samples	#attributes	Imbalance ratio
glass1	214	9	1.82
haberman	306	3	2.78
ecoli1	336	7	3.36
new-thyroid2	215	5	5.14
yeast3	1484	8	8.11
ecoli3	336	7	8.19
glass2	214	9	10.39
yeast1_7	459	8	13.87
ecoli4	336	7	15.75
abalone9_18	731	8	16.39
glass5	214	9	23.42
yeast5	1484	8	32.89

Table 4 *G-mean* comparison results with several algorithms

Dataset	Non-ensemble		Vote-based ensemble		DE-based ensemble	
	(*C*, *L*)	*G-mean* (%)	(*C*, *L*)	*G-mean* (%)	(*C*, *L*)	*G-mean* (%)
glass1	(2^{16}, 780)	73.46	(2^{30}, 590)	74.32	(2^{18}, 390)	**77.72**
haberman	(2^{-6}, 350)	60.37	(2^{12}, 140)	**63.10**	(2^{28}, 540)	62.68
ecoli1	(2^{-2}, 270)	89.21	(2^{40}, 890)	89.72	(2^{0}, 390)	**91.39**
new-thyroid2	(2^{16}, 290)	**99.47**	(2^{10}, 340)	**99.47**	(2^{32}, 560)	99.24
yeast3	(2^{8}, 810)	92.11	(2^{4}, 270)	94.25	(2^{2}, 490)	**94.57**
ecoli3	(2^{20}, 40)	88.59	(2^{10}, 10)	88.68	(2^{18}, 40)	**89.50**
glass2	(2^{12}, 380)	79.02	(2^{8},170)	86.45	(2^{16}, 350)	**87.51**
yeast1_7	(2^{10}, 180)	76.01	(2^{20}, 370)	**78.95**	(2^{38}, 20)	78.94
ecoli4	(2^{24}, 190)	96.01	(2^{8}, 750)	96.33	(2^{14}, 310)	**96.77**
abalone9_18	(2^{24}, 260)	87.89	(2^{4}, 120)	89.24	(2^{16}, 330)	**90.13**
glass5	(2^{26}, 390)	94.38	(2^{18}, 570)	**94.55**	(2^{12}, 260)	**94.55**
yeast5	(2^{16}, 480)	93.27	(2^{12}, 330)	94.51	(2^{28}, 430)	**94.59**
Average	–	85.82	–	87.46	–	**88.13**

In the WELM algorithm, there are two parameters that need to be optimized. One is the trade-off constant *C*, and the other is the number of hidden nodes *L*. In this paper, a grid search of *C* $\{2^{-10}, 2^{-8}, \ldots, 2^{38}, 2^{40}\}$ and *L* {10, 20, …, 890, 900} is used to find the optimal result.

The proposed DE-based ensemble algorithm and two compared algorithms are tested on 12 datasets. The average classification *G-mean*s are recorded as well as the corresponding optimal *C* and *L* in Table 4. The table also shows the average results of all datasets. The best performance result per dataset is shown in bold type. Observing the average results in Table 4, ensemble approaches show an obvious advantage over the non-ensemble approach and help to improve the classification performance. Non-ensemble approach obtains the lowest *G-mean*s on 11 out of 12 datasets, only except for *new-thyroid2* dataset.

We also could find that the DE-based ensemble evidently outperforms the other two compared algorithms in terms of *G-mean*s on 8 out of 12 datasets, except for *haberman*, *new-thyroid2*, *yeast1_7*, and *glass5* datasets. Only on *new-thyroid2* dataset, non-ensemble and vote-based ensemble perform slightly better than DE-based ensemble. The reason lies that DE-based ensemble may overtrain in the small samples and lead to lower classification performance. It also can be observed that the *G-mean*s from the non-ensemble and vote-based ensemble show no significance difference on *new-thyroid2* dataset. Moreover, vote-based ensemble obviously outperforms DE-based ensemble and non-ensemble on *haberman* and *yeast1_7* datasets. It can be found through careful observation that the DE-based ensemble has not improved significantly the performance in terms of *G-mean* on *ecoli4*, *glass5* and *yeast5* datasets, while it slightly outperforms vote-based ensemble and non-ensemble.

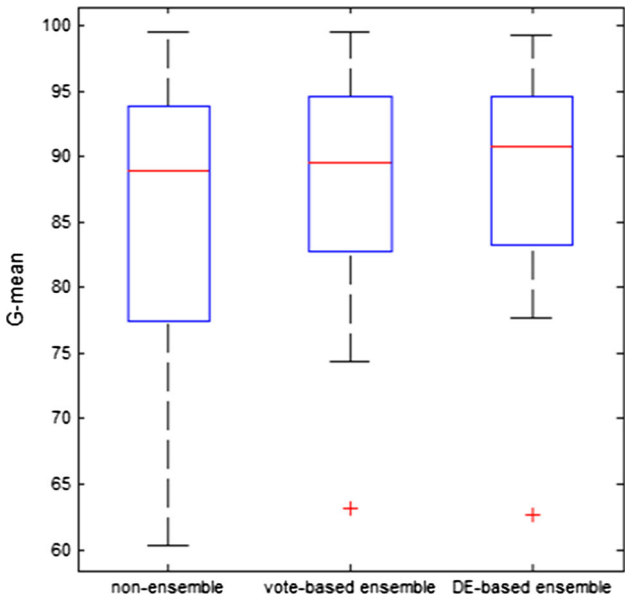

Fig. 1 The *box* plots of experimental results

Among all the three comparison algorithms, DE-based ensemble obtains the best performance on average *G-mean*s which reaches 88.13, while the vote-based ensemble and the non-ensemble are 87.46 and 85.82, respectively. The results indicate that our proposed approach shows a strong competitiveness in terms of *G-mean* against two other approaches.

We also use the box plot to show the experimental results in Fig. 1. The box generated by the DE-based ensemble is shorter than the boxes generated by the non-ensemble and vote-based ensemble. In other words, dispersion degree of DE-based ensemble is relatively low. It should be noticed that the box plots of DE-based ensemble

and vote-based ensemble consider the *G-means* of *haberman* dataset as an exception. The box plots show that DE-based ensemble is more robust than the other two algorithms.

6 Conclusions

Imbalanced data are common in real applications, and the class imbalance learning problem has been widely studied. Ensemble approach is one of the efficient methods to solve class imbalance learning problem. By means of fast learning speed and good generalization performance of ELM, this paper presents a novel ensemble approach on WELM for the CIL problem. In the proposed ensemble strategy, the weights of base learners are optimized by DE algorithm. The experimental results on several datasets have demonstrated that the performance of WELM in terms of *G-mean* is significantly improved after ensemble strategy based on differential evaluation is applied.

Acknowledgments This work is partly supported by National Natural Science Foundation of China (No. 61373127) and the State Key Laboratory for Novel Software Technology (Nanjing University) of China (No. KFKT2015B16).

References

1. Zhang D, Islam MM, Lu G (2012) A review on automatic image annotation techniques. Pattern Recogn 45(1):346–362
2. Garcia-Pedrajas N, Perez-Rodriguez J, Garcia-Pedrajas MD, Ortiz-Boyer D, Fyfe C (2012) Class imbalance methods for translation initiation site recognition in DNA sequences. Knowl Based Syst 25(1):22–34
3. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
4. Tang Y, Zhang YQ, Chawla NV, Krasser S (2009) SVMs modeling for highly imbalanced classification. IEEE Trans Syst Man Cybernet B 39(1):281–288
5. Liu X, Wu J, Zhou Z (2009) Exploratory undersampling for class-imbalance learning. IEEE Trans Syst Man Cybernet B 39(2): 539–550
6. Zhou Z, Liu X (2006) Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Trans Knowl Data Eng 18(1):63–77
7. Sun Y, Kamel MS, Wong AKC, Wang Y (2007) Cost-sensitive boosting for classification of imbalanced data. Pattern Recogn 40(12):3358–3378
8. 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 C Appl Rev 42(4):463–484
9. Xue X, Yao M, Wu Z, Yang J (2014) Genetic ensemble of extreme learning machine. Neurocomputing 129:175–184
10. Zhu QY, Qin AK, Suganthan PN, Huang GB (2005) Evolutionary extreme learning machine. Pattern Recogn 38(10):1759–1763
11. Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359
12. Price K, Storn R, Lampinen J (2005) Differential evolution: a practical approach for global optimization". Springer, Berlin
13. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
14. Zong W, Huang GB, Chen YQ (2013) Weighted extreme learning machine for imbalance learning. Neurocomputing 101:229–242
15. Li K, Kong X, Lu Z, Liu W, Yin J (2014) Boosting weighted ELM for imbalanced learning. Neurocomputing 128:15–21
16. Yu Q, Heeswijk M, Miche Y, Nian R, He B, Séverin E, Lendasse A (2014) Ensemble delta test-extreme learning machine (DT-ELM) for regression. Neurocomputing 129:153–158
17. Mirza B, Lin Z, Toh KA (2013) Weighted online sequential extreme learning machine for class imbalance learning. Neural Process Lett 38(3):465–486
18. Huang GB, Song S, Gupta J, Wu C (2014) Semi-supervised and unsupervised extreme learning machines. IEEE Trans Cybernet 44(12):2405–2417
19. Xia SX, Meng FR, Liu B, Zhou Y (2015) A kernel clustering-based possibilistic fuzzy extreme learning machine for class imbalance learning. Cognit Comput 7(1):74–85
20. Mao W, Wang J, Xue Z (2016) An ELM-based model with sparse-weighting strategy for sequential data imbalance problem. Int J Mach Learn Cybernet. doi:10.1007/s13042-016-0509-z
21. Mirza B, Lin Z, Liu N (2015) Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift. Neurocomputing 149:316–329
22. Cao JW, Lin ZP, Huang GB, Liu N (2012) Voting based extreme learning machine. Inf Sci 185(1):66–77
23. Feng G, Qian Z, Zhang X (2012) Evolutionary selection extreme learning machine optimization for regression. Soft Comput 16(9):1485–1491
24. Cao JW, Lin ZP, Huang GB (2012) Self-adaptive evolutionary extreme learning machine. Neural Process Lett 36:285–305
25. Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A (2010) OP-ELM: optimally pruned extreme learning machine. IEEE Trans Neural Netw 21(1):158–162
26. Liang NY, Saratchandran P, Huang GB, Sundararajan N (2006) Classification of mental tasks from EEG signals using extreme learning machine. Int J Neural Syst 16(1):29–38
27. Huang GB, Chen L (2007) Convex incremental extreme learning machine. Neurocomputing 70(16):3056–3062
28. Huang G, Huang GB, Song S, You K (2015) Trends in extreme learning machines: a review. Neural Netw 61:32–48
29. Mukherjee R, Patra GR, Kundu R, Das S (2014) Cluster-based differential evolution with crowding archive for niching in dynamic environments. Inf Sci 267:58–82
30. KEEL dataset repository. http://sci2s.ugr.es/keel/imbalanced.php