Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

G-mean based extreme learning machine for imbalance learning

JongHyok Ri [a,*], Hun Kim [b]

[a] *Institute of Information Technology, Kim Il Sung University, Pyongyang, Democratic People's Republic of Korea*
[b] *School of Information Science, Kim Il Sung University, Pyongyang, Democratic People's Republic of Korea*

A R T I C L E   I N F O

Article history:
Available online xxxx

Keywords:
Extreme learning machine
Imbalance learning
Optimization
G-mean

A B S T R A C T

Although extreme learning machine (ELM) provides better generalization performance at a much faster learning speed than traditional learning algorithms, the classical ELM can not obtain ideal results for the imbalanced data problem. In this paper, in order to conquer the learning capability of the classical ELM for an imbalance data learning, we define a new cost function of ELM optimization problem based on G-mean widely used as evaluation metric in imbalance data learning. We perform experiments on standard classification datasets which consist of 58 binary datasets and 11 multi-class datasets with the different degrees of the imbalance ratio. Experimental results show that proposed algorithm can improve the classification performance significantly compared with other state-of-the-art methods. We also demonstrate that our proposed algorithm can achieve high accuracy in representation learning by performing experiments on YouTube-8M with feature representation from convolutional neural networks. Statistical results indicate that the proposed approach not only outperforms the classical ELM, but also yields better or at least competitive results compared with several state-of-the-art class imbalance learning approaches.

1. Introduction

Extreme learning machine (ELM) [1–4] is a fast and simple learning algorithm for single-hidden layer feedforward neural (SLFN) networks training. In the past decade, ELM has been attracted attention from various real-world fields [5–7].

In general, most of the data we contact in real applications is imbalanced, and data belonging to minority class tend to have greater significance. In recent years, a number of methods have been developed to deal with class imbalance problem, which falls into four groups: data sampling; cost-sensitive learning; decision boundary moving and ensemble learning. Data sampling redistributes the training data by sampling, which allows classifiers to perform in the conventional way. The most used approaches are undersampling and oversampling, either eliminating the majority class instances or increasing the minority class instances [8–10]. The cost-sensitive learning can work on the data level, algorithm level, or both. It takes the misclassification cost into consideration and assigns a higher cost to misclassified positive samples [11,12]. The decision boundary moving [13,14] attempt to artificially move the decision boundary towards the majority class by using disparate costs. The ensemble learning trains multiple classifiers for the same task and combines their predictions to classify new instances. The popular approaches include bagging-family-based, boosting-family-based, and random forests-family-based learning methods [15,16]. When applying conventional learning algorithms [17,18] for an imbalance data, the minority class data are easily misclassified.

Similar to other classifiers, although the ELM algorithm works effectively with balanced datasets, when classifying data with imbalance class distribution, it can obtain undesirable model that has a natural tendency to favor the majority class and a low performance on the minority class. Because classical ELM assumes balanced class distribution or equal misclassification cost for the different categories of classes from the viewpoint of the optimization.

In order to improve the performance of ELM algorithm for an imbalance data, several scholars have proposed their research results over the past few years. Zong et al. [19] proposed weighted ELM to overcome the weakness of the classic ELM for an imbalance data problem. Weighted ELM assigned the extra weight to each sample, which is simply calculated by using the number of the sample in class. So that weighted ELM strengthen the impact of the minority class while weaken the relative impact of the majority class. However weighted ELM was rely on empirically weighting schemes which only were designed according to the element number of each class, thus the global optima can not be guaranteed.

* Corresponding author.
E-mail address: tech2@ryongnamsan.edu.kp (J. Ri).

https://doi.org/10.1016/j.dsp.2019.102637
1051-2004/© 2019 Elsevier Inc. All rights reserved.

In order to improve the classification performance of weighted ELM, boosting weighted ELM [20] has been proposed. The key essence of the boosting weighted ELM is to introduce the boosting method to obtain better weighting schemes in weighted ELM. The boosting based method tries to determine the optimal weight matrix by an Adaboost algorithm [21]. In Yu et al. [13], authors analyzed the reason of the damage caused by class imbalance for ELM in theory and presented an optimal decision output compensation-based ELM (ODOC-ELM) for improving the classification performance of ELM in the scenario of the class imbalance. Also they compared their research result with state-of-the-art methods by experiments.

In this article, in order to improve the weakness of the classical ELM for an imbalance data learning, we define the newly cost function of ELM optimization problem based on G-mean widely used as evaluation metric in imbalance data learning and propose new ELM algorithm based on this cost function. In our approach, the cost function try to minimize the product of training errors of each class instead of the total sum of the training errors and introduce to the logarithmic function for convenience of later expansion. We proved effectiveness of our machine learning algorithm by performing experiments on standard classification datasets which consist of 58 binary datasets and 11 multi-class datasets, as well as Youtube-8M datasets. and compared our learning algorithms with state-of-the-art method by using experimental results of Yu et al. [13]. Statistical results indicate that the proposed approach not only outperforms the classical ELM, but also yields better or at least competitive results compared with several state-of-the-art class imbalance learning approaches.

The contributions of our approach are given as follows,

- We define a new cost function of ELM optimization problem based on G-mean widely used as evaluation metric in imbalance data learning.
- We propose a new ELM algorithm based on this cost function in order to improve the weakness of classical ELM for the imbalance data learning.
- Our approach can achieve significant improvement in classification performance compared with other state-of-the-art methods on various imbalance datasets.

The paper is organized as the following. Section 2 introduces the related work of ELM on imbalance data learning. Section 3 presents the proposed method. Section 4 reports the experimental results and performance analysis. Finally, the conclusions are summarized in Section 5.

2. Previous work

2.1. ELM

ELM [1,2] was originally proposed for the single-hidden layer feedforward neural networks and then extended to "generalized" single-hidden layer feedforward networks (SLFNs) where the hidden layer does not require tuning [3].

The main feature of ELM is that all the hidden neuron parameters are randomly generated and the output weights are analytically decided by the Moore-Penrose generalized inverse [3].

Given N training samples $(\boldsymbol{x}_i, \boldsymbol{t}_i)_{i=1}^N$ where $\boldsymbol{x}_i$ is the feature vector of the ith sample and $\boldsymbol{t}_i$ is the target vector. If a SLFN with L hidden nodes can approximate the data with zero error, then the following equation holds:

$$\boldsymbol{t}_i = \sum_{j=1}^{L} \boldsymbol{\beta}_j f(\boldsymbol{x}_i, \boldsymbol{a}_j, b_j), \quad i = 1, \dots, N \tag{1}$$

where $\boldsymbol{\beta}_j$ is the weight vector connecting the jth hidden node to the output nodes. $f(\boldsymbol{x}_i, \boldsymbol{a}_j, b_j)$ is the activation function, where $\boldsymbol{a}_j$ is the weight vector connecting the jth hidden node to the input nodes, b_j is the bias vector of the jth hidden node.

The Eq. (1) could be rewritten in matrix form as:

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T} \tag{2}$$

where $\boldsymbol{H}$ is the hidden layer output matrix and defined as:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{h}(\boldsymbol{x}_N) \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{x}_1, \boldsymbol{a}_1, b_1) & \cdots & f(\boldsymbol{x}_1, \boldsymbol{a}_L, b_L) \\ \vdots & \cdots & \vdots \\ f(\boldsymbol{x}_N, \boldsymbol{a}_1, b_1) & \cdots & f(\boldsymbol{x}_N, \boldsymbol{a}_L, b_L) \end{bmatrix}_{N \times L} \tag{3}$$

$\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_L]^T$ and $\boldsymbol{T} = [\boldsymbol{t}_1, \dots, \boldsymbol{t}_N]^T$

The output weight matrix $\boldsymbol{\beta}$ could be estimated analytically by the minimum norm least-square solution:

$$\boldsymbol{\beta} = \boldsymbol{H}^{\dagger}\boldsymbol{T} \tag{4}$$

where $\boldsymbol{H}^{\dagger}$ is the Moore-Penrose "generalized" inverse [3,22–24] of the hidden layer output matrix $\boldsymbol{H}$.

2.2. The variants of ELM on imbalanced learning

In ELM classifier, the separating boundary is supposed to be pushed toward the side of the minority class for an imbalance data, so that the minority classes is easy misclassified. In order to resolve this issue, weighted ELM [19] is proposed.

Weighted ELM can be regarded as a cost sensitive learning method, as it offers a larger penalty to the training errors of the minority instances than to those of the majority ones. In fact, each training sample is assigned with an extra weight. Mathematically, a $N \times N$ diagonal matrix $\boldsymbol{W}$ associated with every training sample $\boldsymbol{x}_i$ is defined. Usually if $\boldsymbol{x}_i$ comes from a minority class, the associated weight w_{ii} is relatively larger than samples from a majority class. Therefore, the impact of the minority class was strengthened while the relative impact of the majority class was weakened. Considering the diagonal weight matrix $\boldsymbol{W}$, the optimization formula of ELM could be revised [19] as

$$\begin{cases} \boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} L_{WELM}(\boldsymbol{\beta}) \\ L_{WELM} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + c\frac{1}{2}\sum_{i=1}^{N}(w_{ii} \times \|\boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} - \boldsymbol{t}_i\|^2) \end{cases} \tag{5}$$

where c is the trade-off regularization parameter between the minimization of the training errors and the maximization of the marginal distance. According to the KKT theorem [24], the solution to Eq. (5) is

$$\boldsymbol{\beta} = \boldsymbol{H}^{\dagger}\boldsymbol{T} = \begin{cases} \boldsymbol{H}^T\left(\frac{\boldsymbol{I}}{C} + \boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T\right)^{-1}\boldsymbol{W}\boldsymbol{T} & N < L \\ \left(\frac{\boldsymbol{I}}{C} + \boldsymbol{H}^T\boldsymbol{W}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^T\boldsymbol{W}\boldsymbol{T} & N \geq L \end{cases} \tag{6}$$

They empirically proposed two weighting schemes;

$$\begin{aligned} W_1 &: \quad w_{ii} = \frac{1}{\#n_i} \\ W_2 &: \quad w_{ii} = \begin{cases} \frac{0.618}{\#n_i} & \text{if } \#n_i > AVG(\#n_i) \\ \frac{1}{\#n_i} & \text{if } \#n_i \leq AVG(\#n_i) \end{cases} \end{aligned} \tag{7}$$

where $\#n_i$ is the number of the samples belonging to class n_i, $AVG(\#n_i)$ represents the average number of the samples for all classes.

The weighted ELM improved the efficiency of ELM algorithm greatly for an imbalance data. The classification error of the class with less elements was reduced by setting a unequal cost distribution for each class with the weight matrix. However, their approach was rely on empirically weighting schemes, which only were designed according to the element number of each class. Thus it can be also called experienced WELM. In this paper, these two WELM classifiers are referred to as WELM W1 and WELM W2.

To overcome this shortcoming of the experienced WELM, the boosting based weighted ELM (BWELM) [20] was proposed. BWELM tries to determine the optimal weight matrix by an Adaboost algorithm. In Yu et al. [13], authors analyzed the reason of the damage caused by class imbalance for ELM in theory and presented a decision output compensation algorithm called ODOC-ELM, which embeds an optimization procedure within itself. The optimal compensation thresholds are automatically found by the golden section search algorithm for binary-class problems, and by particle swarm optimization for multi-class problems. Also they compared their research result with state-of-the-art methods by experiments.

In this paper, we present a new ELM algorithm based on G-mean metric for an imbalance learning problem.

3. Our approach

In this section, we describe in detail the proposed novel ELM classifier based on G-mean widely used as evaluation metric in imbalance data learning.

G-mean, as a conventional evaluation metrics in the case of the imbalance data learning, is the geometric mean of the recall values of all M classes. Given training dataset $\mathbf{\Phi} = \{(\boldsymbol{x}_i, \boldsymbol{t}_i), \boldsymbol{t}_i \in R^M, i = 1, 2, \dots, N\}$, there are m classes in $\mathbf{\Phi}$. Where $\boldsymbol{t}_i \in R^M$ is a target label and $\boldsymbol{x}_i$ is a data vector. Then G-mean is defined as follows,

$$\text{G-mean} = \left(\prod_{j=1}^{M} \frac{v_j}{V_j}\right)^{\frac{1}{M}} \tag{8}$$

where v_j is the number of elements correctly classified among elements of class j and V_j is the number of elements belonging in class j. According to the category, we can write the given training dataset $\mathbf{\Phi}$ as follows,

$$\mathbf{\Phi} = \bigcup_{j=1}^{M} \Phi_j \tag{9}$$

where Φ_j is a j-th class set among training dataset $\mathbf{\Phi}$. Now the training error of each class set Φ_j in ELM is defined as follows,

$$\Xi_j = \sum_{\boldsymbol{x} \in \Phi_j} \|\boldsymbol{\xi}(\boldsymbol{x})\|^2, \quad j = 1, \dots, M \tag{10}$$

where $\boldsymbol{\xi}(\boldsymbol{x})$ is training error of $\boldsymbol{x}$ belonging in j-th class in ELM classifier, it is defined as follows,

$$\boldsymbol{\xi}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta} - \boldsymbol{t}(\boldsymbol{x}) \tag{11}$$

where $\boldsymbol{t}(\boldsymbol{x})$ is corresponding class label of $\boldsymbol{x}$ and $\boldsymbol{h}(\boldsymbol{x})$ is hidden layer output obtained using Eq. (3).

Now the optimization formula of the standard ELM in the optimization view could be revised as.

$$Minimize: L_{ELM} = \tfrac{1}{2}\|\boldsymbol{\beta}\|^2 + c\tfrac{1}{2}\sum_{j=1}^{M} \Xi_j \tag{12}$$

As can be seen from above cost function, original ELM tried to minimize the sum of the training errors and to maximize the marginal distance between classes. Pondering the definition of G-mean widely used as evaluation metric, we will define a new cost function to improve the performance of ELM in imbalance data learning.

In general, it is clear that the smaller the training error, the better the classification accuracy. Considering the definition of Ξ_j, it is easy to get as follows.

$$\frac{v_j}{V_j} \propto \frac{1}{\Xi_j}, \quad j = 1, \dots, M \tag{13}$$

Then, according to the definition of G-mean, we also have as follows.

$$\begin{aligned}
&\left(\prod_{j=1}^{M} \frac{v_j}{V_j}\right)^{\frac{1}{M}} \propto \left(\prod_{j=1}^{M} \frac{1}{\Xi_j}\right)^{\frac{1}{M}} \\
&\left(\prod_{j=1}^{M} \frac{v_j}{V_j}\right)^{\frac{1}{M}} \propto \prod_{j=1}^{M} \frac{1}{\Xi_j} \\
&\left(\prod_{j=1}^{M} \frac{v_j}{V_j}\right)^{\frac{1}{M}} \propto \frac{1}{\prod_{j=1}^{M} \Xi_j} \\
&\text{G-mean} \propto \frac{1}{\prod_{j=1}^{M} \Xi_j}
\end{aligned} \tag{14}$$

The above equation show that the smaller the product of training errors of each class for imbalanced data learning, the bigger the value of G-mean. In other words, to maximize G-mean is equivalent to minimize the product of training errors of each class for imbalanced data learning. Thus we have as follows,

$$max\ \text{G-mean} \propto min \prod_{j=1}^{M} \Xi_j \tag{15}$$

For convenience of later expansion, we introduce to the logarithmic function again.

$$max\ \text{G-mean} \propto min \log \prod_{j=1}^{M} \Xi_j \tag{16}$$

Now we will define a new cost function of ELM based on G-mean as follows.

$$Minimize: L_{GELM} = \tfrac{1}{2}\|\boldsymbol{\beta}\|^2 + c\tfrac{1}{2}\log \prod_{j=1}^{M} \Xi_j \tag{17}$$

By using Eq. (10), we also have as follows.

$$Minimize: L_{GELM} = \tfrac{1}{2}\|\boldsymbol{\beta}\|^2 + c\tfrac{1}{2}\log \prod_{j=1}^{M} \Big(\sum_{\boldsymbol{x} \in \Phi_j} \|\boldsymbol{\xi}(\boldsymbol{x})\|^2\Big) \tag{18}$$

By property of the logarithmic function, it can be written as follows.

$$Minimize: L_{GELM} = \tfrac{1}{2}\|\boldsymbol{\beta}\|^2 + c\tfrac{1}{2}\sum_{j=1}^{M} \log\Big(\sum_{\boldsymbol{x} \in \Phi_j} \|\boldsymbol{\xi}(\boldsymbol{x})\|^2\Big) \tag{19}$$

By Eq. (11),

$$\begin{aligned}
&Minimize: \\
&L_{GELM} = \tfrac{1}{2}\|\boldsymbol{\beta}\|^2 + c\tfrac{1}{2}\sum_{j=1}^{M} \log\Big(\sum_{\boldsymbol{x} \in \Phi_j} \|\boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta} - \boldsymbol{t}(\boldsymbol{x})\|^2\Big)
\end{aligned} \tag{20}$$

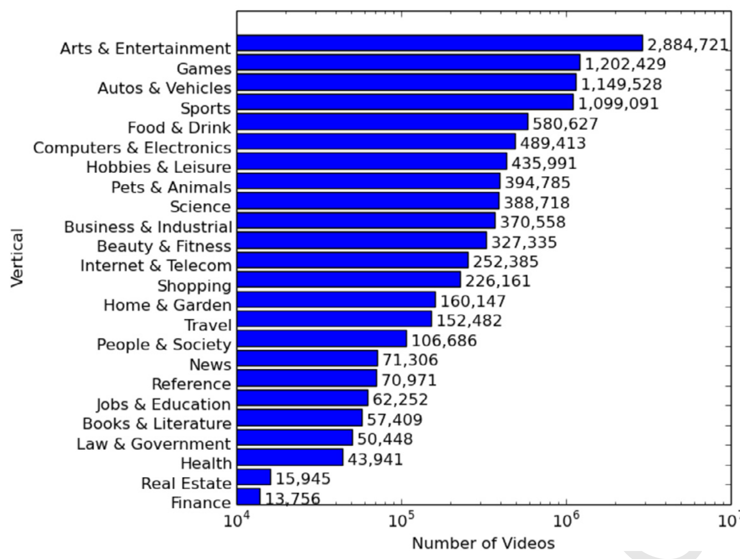

Fig. 1. Number of train videos in log-scale per top-level category.

Finally, the new optimization problem we define is as follows.

$$\begin{cases} \boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} L_{GELM}(\boldsymbol{\beta}) \\ L_{GELM} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + c\frac{1}{2}\sum_{j=1}^{M}\log(\sum_{\boldsymbol{x}\in\Phi_j}\|\boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta}-\boldsymbol{t}(\boldsymbol{x})\|^2) \end{cases} \tag{21}$$

In order to obtain the minimum of L_{GELM} with respect to $\boldsymbol{\beta}$, we will resort to an iterative optimization algorithms such as gradient descent, L-BFGS [25–29]. Taking derivatives, one can show that the derivative is;

$$\nabla_{\boldsymbol{\beta}} L_{GELM} = \boldsymbol{\beta} + c\sum_{j=1}^{M}\frac{\sum_{\boldsymbol{x}\in\Phi_j}\boldsymbol{h}^T(\boldsymbol{x})(\boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta}-\boldsymbol{t}(\boldsymbol{x}))}{\sum_{\boldsymbol{x}\in\Phi_j}(\boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta}-\boldsymbol{t}(\boldsymbol{x}))^2} \tag{22}$$

Algorithm 1 GELM.

```
1: Input training set $\boldsymbol{\Phi} = \{(\boldsymbol{x}_i, \boldsymbol{t}_i), \boldsymbol{t}_i \in R^M, i = 1, 2, \dots, N\}$ and activation function
   of hidden layer $f$.
2: All data value $\boldsymbol{x}_i$ of training set $\Phi$ is normalized in [-1,1] and randomly assign
   input parameter $\boldsymbol{a}_k, b_k, k = 1, \dots, L$.
3: Compute hidden layer outputs $\boldsymbol{h}(\boldsymbol{x}_i)$ of all data value $\boldsymbol{x}_i$ using Eq. (3).
4: Seek an optimal solution $\boldsymbol{\beta}^*$ of GELM using iterative optimization algorithm
   with Eq. (22).
5: Return Output weight $\boldsymbol{\beta}^*$.
```

The processing steps followed for the parameter calculation of our Approach are illustrated in Algorithm 1. The proposed Algorithm is applicable to both binary and multi-class problems.

4. Experiments

In this section, we report and discuss the results of comparable experiments to evaluate the classification capability of the proposed GELM on imbalance learning problem.

4.1. Dataset specification

4.1.1. Standard classification datasets

We compared the proposed GELM with other algorithms on 58 binary-class datasets and 11 multi-class datasets randomly acquired from the KEEL dataset repository [30] which have different degrees of imbalance. The imbalance ratio (IR) of binary datasets selected from KEEL data repository varies from 1.82 to 85.88, the imbalance ratio (IR) of multi-class datasets varies from 1.5 to 853.

Table 1
Selected imbalanced datasets from Youtube-8M database.

Datasets	#Attributes	#Classes	#Instances	IR
TrainData	1024	10	3160	8.34
TestData	1024	10	1580	8.34

4.1.2. Youtube-8M database

Youtube-8M dataset [31] contains around 8 million Youtube videos. Each video is annotated with one or multiple tags with a total of 4716 classes. All 4716 classes can be grouped into 24 top-level vertical categories. In Fig. 1, we show histograms with the number of training videos in each top-level vertical.

In this dataset, visual and audio features are pre-extracted. Visual features are obtained by the Google Inception CNN, which is pre-trained on the ImageNet [32], those features are then reduced by the PCA-compression into a 1024 dimensional vector. The audio features are extracted from a pre-trained VGG [33] network. In the official split, the dataset is divided into three parts: 70% for training, 20% for validation, and 10% for testing.

In our experiment, we randomly selected 4740 videos, which have unique top-level vertical categories with visual features from the official dataset, to validate our algorithm on video classification problem. The details of selected datasets are shown in Table 1.

4.2. Setup

We first compare the performance of the proposed GELM with four state-of-the-art methods, i.e., the Softmax classifier [34], the classical ELM classifier [1], the experienced WELM [19] and the boosting weighted ELM [20] on standard classification datasets (58 binary-class datasets and 11 multi-class datasets) and a subset of Youtube-8M database. The results are averaged over 10 runs.

In Yu et al. [13], the performances of SMOTE-ELM [8], RWOS-ELM [9], SVM-OTHR [14] and ODOC-ELM [13] were already evaluated on 27 binary-class datasets and 9 multi-class datasets among standard classification datasets (58 binary-class datasets and 11 multi-class datasets). Therefore, we also compare GELM with SMOTE-ELM [8], RWOS-ELM [9], SVM-OTHR [14], ODOC-ELM [13] by using experimental results of Yu et al. [13]. SMOTE-ELM [8] is an original ELM algorithm with preprocessing the training data using SMOTE technology, where the number of neighbors k in SMOTE is assigned as the default value 5. RWOS-ELM [9] is an original ELM algorithm with preprocessing the training data using random walk oversampling technology. SVM-OTHR [14] is a support vector machine-based optimized decision threshold adjustment strategy.

All experiments are implemented in Matlab 2016b environment. In ELM theory, there are many choices for the feature mapping. In our experiments, we use the sigmoid additive based feature mapping function [4] on the hidden layer, which is a popular choice by researchers. There are two parameters to tune for ELM, i.e. the trade-off factor C and the number of hidden nodes l. In our experiments, similar to experiment setting of Yu et al. [13], the grid search ranges based on five-fold cross-validation are set as follows,

$$C \in \{2^{-20}, 2^{-18}, \ldots, 2^{18}, 2^{20}\}$$
$$l \in \{10, 20, \ldots, 190, 200\}$$

In order to obtain the minimum of GELM, we will use L-BFGS. The search range of λ for Softmax classifier is set as $\{-8, -7, -6, \ldots, 1, 2, 3\}$.

For performance estimation metrics, we adopted G-mean to compare the performance of various learning algorithms, and then provided the experimental results in the form of mean ± standard deviation.

4.3. Experimental results on standard classification problem datasets

The comparable experimental results of these six methods (Softmax, ELM, WELM W1, WELM W2, BWELM, and GELM) under 58 binary imbalance datasets and 11 multi-class imbalance datasets are given in Table 2 and Table 3 respectively. The results in both tables show our method can perform much better than all

Table 2
Performance results on 58 binary imbalance datasets.

Datasets	G-mean					
	Softmax	ELM	WELM W1	WELM W2	BWELM	GELM
cleveland0vs4	0.7234 ± 0.1139	0.9383 ± 0.1294	0.9769 ± 0.0070	0.9840 ± 0.0226	0.9836 ± 0.0142	**0.9872 ± 0.0209**
dermatology6	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
ecoli0137vs26	0.7401 ± 0.1336	0.7414 ± 0.1334	**0.9926 ± 0.0166**	0.9665 ± 0.0207	0.9845 ± 0.0124	**0.9926 ± 0.0120**
ecoli0146vs5	0.7608 ± 0.1502	0.9127 ± 0.1282	0.9428 ± 0.0741	0.9372 ± 0.0732	0.9477 ± 0.0617	**0.9484 ± 0.0732**
ecoli01vs235	0.7892 ± 0.1040	0.7752 ± 0.1322	0.8719 ± 0.0339	0.7375 ± 0.0612	**0.9649 ± 0.0462**	0.9397 ± 0.0648
ecoli01vs5	0.8273 ± 0.1235	0.6878 ± 0.1212	0.9169 ± 0.0562	0.8372 ± 0.0652	0.9242 ± 0.0340	**0.9448 ± 0.0591**
ecoli0234vs5	0.8567 ± 0.1073	0.9118 ± 0.1275	0.9479 ± 0.1089	0.9377 ± 0.1238	0.9272 ± 0.1252	**0.9574 ± 0.0955**
ecoli0267vs35	0.8067 ± 0.1822	0.9253 ± 0.0692	0.9232 ± 0.0550	0.9265 ± 0.0611	**0.9372 ± 0.0222**	0.9273 ± 0.0720
ecoli0346vs5	0.8583 ± 0.0993	0.9196 ± 0.0734	0.9649 ± 0.0581	0.9448 ± 0.0757	0.9589 ± 0.0249	**0.9795 ± 0.0566**
ecoli034vs5	0.8508 ± 0.0430	0.9146 ± 0.1297	0.9464 ± 0.0734	0.9412 ± 0.0745	0.9440 ± 0.0768	**0.9478 ± 0.0741**
ecoli067vs35	0.8178 ± 0.1628	0.9054 ± 0.1593	0.9298 ± 0.0696	0.9224 ± 0.0788	0.9270 ± 0.0827	**0.9422 ± 0.0535**
ecoli067vs5	0.8040 ± 0.2089	0.9388 ± 0.0682	0.9413 ± 0.0695	0.9476 ± 0.0496	0.9497 ± 0.0229	**0.9559 ± 0.0393**
ecoli3	0.6730 ± 0.1562	0.6717 ± 0.1253	0.8727 ± 0.0381	0.8710 ± 0.0171	0.8898 ± 0.0413	**0.9261 ± 0.0407**
ecoli4	0.9434 ± 0.0739	0.8164 ± 0.0734	0.9636 ± 0.0105	0.9339 ± 0.0128	0.9491 ± 0.0056	**0.9872 ± 0.0133**
glass04vs5	0.9940 ± 0.0134	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
glass06vs5	0.7844 ± 0.1398	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
glass0	0.7451 ± 0.0861	**0.8537 ± 0.0433**	0.8359 ± 0.0435	0.8016 ± 0.0357	0.7557 ± 0.0379	0.7805 ± 0.0321
glass1	0.3751 ± 0.1004	0.6687 ± 0.0487	0.7101 ± 0.0569	0.6813 ± 0.0783	0.7250 ± 0.0695	**0.7498 ± 0.0552**
glass2	0.0000 ± 0.0000	0.6753 ± 0.1036	**0.9289 ± 0.0122**	0.9067 ± 0.0217	0.8831 ± 0.0431	0.9275 ± 0.0464
glass4	0.4379 ± 0.1106	0.9173 ± 0.0180	**0.9398 ± 0.0166**	0.9273 ± 0.0156	0.9349 ± 0.0113	0.9351 ± 0.0431
glass5	0.9851 ± 0.0206	**0.9975 ± 0.0055**	**0.9975 ± 0.0055**	**0.9975 ± 0.0055**	**0.9975 ± 0.0055**	0.9806 ± 0.0545
glass6	0.8870 ± 0.0795	0.9301 ± 0.0445	0.9426 ± 0.0390	0.9309 ± 0.0445	0.9563 ± 0.0169	**0.9604 ± 0.0486**
haberman	0.4915 ± 0.0684	0.3795 ± 0.1118	0.6044 ± 0.0705	0.4890 ± 0.0429	0.6258 ± 0.0291	**0.6488 ± 0.0511**
iris0	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
led7digit02456789vs1	0.8907 ± 0.0593	0.9025 ± 0.0847	0.9290 ± 0.0504	0.9040 ± 0.0388	0.9114 ± 0.0551	**0.9468 ± 0.0450**
newthyroid1	0.9403 ± 0.0406	0.9103 ± 0.0406	0.9874 ± 0.0000	0.9755 ± 0.0126	0.9874 ± 0.0000	**1.0000 ± 0.0000**
newthyroid2	0.9514 ± 0.0668	0.9394 ± 0.0644	**1.0000 ± 0.0000**	0.9916 ± 0.0077	**1.0000 ± 0.0000**	0.9972 ± 0.0063
page-blocks13vs4	0.8219 ± 0.0441	**1.0000 ± 0.0000**	0.9955 ± 0.0047	0.9943 ± 0.0056	0.9989 ± 0.0025	0.8916 ± 0.0537
page-blocks0	0.7665 ± 0.0204	0.8687 ± 0.0082	**0.9328 ± 0.0040**	0.9323 ± 0.0041	0.8684 ± 0.0251	0.8398 ± 0.0100
pima	0.7143 ± 0.0466	0.6992 ± 0.0391	0.7541 ± 0.0284	0.6900 ± 0.0125	0.7401 ± 0.0373	**0.7609 ± 0.0350**
poker89vs5	0.5656 ± 0.0740	0.2681 ± 0.2447	0.6129 ± 0.0976	0.5827 ± 0.0756	0.6428 ± 0.0443	**0.6519 ± 0.1244**
poker8vs6	0.5352 ± 0.1195	0.4041 ± 0.3873	0.8390 ± 0.0061	**0.8493 ± 0.0050**	0.8397 ± 0.0008	0.8446 ± 0.1297
shuttle2vs5	0.9998 ± 0.0003	0.9998 ± 0.0003	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
shuttle6vs23	0.9414 ± 0.1310	0.8807 ± 0.0871	0.9380 ± 0.1005	0.9607 ± 0.0300	0.9707 ± 0.0201	**0.9977 ± 0.0051**
shuttlec0vsc4	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
shuttlec2vsc4	0.9414 ± 0.1310	0.9923 ± 0.0764	0.9931 ± 0.1084	0.9915 ± 0.0805	0.9953 ± 0.0420	**1.0000 ± 0.0000**
vehicle1	0.6489 ± 0.0641	0.7555 ± 0.0489	0.8230 ± 0.0140	0.7871 ± 0.0095	0.8117 ± 0.0309	**0.8382 ± 0.0354**
vowel0	0.8241 ± 0.0350	0.9371 ± 0.0681	0.9297 ± 0.0684	0.9177 ± 0.0792	0.9410 ± 0.0890	**0.9493 ± 0.0494**
winequalityred3vs5	0.2828 ± 0.3873	0.4217 ± 0.3849	**0.9349 ± 0.0371**	0.9126 ± 0.0901	0.8940 ± 0.0356	0.9186 ± 0.0944
winequalityred4	0.1234 ± 0.1691	0.2370 ± 0.2218	0.6850 ± 0.0479	0.6688 ± 0.0451	0.6905 ± 0.0625	**0.7311 ± 0.0430**
winequalityred8vs6	0.1155 ± 0.2582	0.5146 ± 0.3105	0.8967 ± 0.0660	0.9023 ± 0.0362	**0.9120 ± 0.0297**	0.7855 ± 0.0761
winequalityred8vs67	0.0000 ± 0.0000	0.2132 ± 0.2933	0.8238 ± 0.0696	**0.8561 ± 0.0495**	0.8464 ± 0.0351	0.7288 ± 0.0767
wisconsin	0.9170 ± 0.0418	0.9616 ± 0.0124	0.9656 ± 0.0052	0.9607 ± 0.0048	0.9637 ± 0.0151	**0.9802 ± 0.0059**
yeast0256vs3789	0.6278 ± 0.0802	0.7359 ± 0.0487	**0.8283 ± 0.0494**	0.7930 ± 0.0437	0.7882 ± 0.0724	0.8111 ± 0.0497
yeast02579vs368	0.8771 ± 0.0705	0.9062 ± 0.0500	0.9075 ± 0.0300	0.9143 ± 0.0232	0.8652 ± 0.0306	**0.9164 ± 0.0348**
yeast0359vs78	0.4600 ± 0.0940	0.5137 ± 0.1038	0.6133 ± 0.0435	0.5997 ± 0.0240	0.7540 ± 0.0661	**0.7851 ± 0.0437**

(*continued on next page*)

Table 2 (*continued*)

Datasets	G-mean					
	Softmax	ELM	WELM W1	WELM W2	BWELM	GELM
yeast05679vs4	0.5818 ± 0.1163	0.7246 ± 0.1605	**0.8798 ± 0.0556**	0.8476 ± 0.0514	0.8420 ± 0.0303	0.8545 ± 0.0434
yeast1289vs7	0.0000 ± 0.0000	0.5671 ± 0.1049	0.7268 ± 0.0257	**0.7458 ± 0.0374**	0.6785 ± 0.0602	0.7367 ± 0.0503
yeast1458vs7	0.0000 ± 0.0000	0.2775 ± 0.2629	**0.7164 ± 0.1111**	0.6630 ± 0.0878	0.6651 ± 0.0580	0.6538 ± 0.0713
yeast1vs7	0.2783 ± 0.2634	0.5502 ± 0.1064	0.7533 ± 0.0422	0.6767 ± 0.0685	0.7644 ± 0.0594	**0.7837 ± 0.0634**
yeast2vs4	0.7999 ± 0.0763	0.8902 ± 0.0711	0.9624 ± 0.0272	**0.9648 ± 0.0214**	0.9305 ± 0.0380	0.9093 ± 0.0303
yeast2vs8	0.6975 ± 0.1301	0.8025 ± 0.0870	0.8246 ± 0.0689	0.8197 ± 0.0721	**0.8622 ± 0.0974**	0.8207 ± 0.0672
yeast1	0.3904 ± 0.0501	0.5141 ± 0.0527	0.6315 ± 0.0241	0.6580 ± 0.0240	0.6328 ± 0.0426	**0.6726 ± 0.0247**
yeast3	0.7580 ± 0.0771	0.7721 ± 0.0268	0.9135 ± 0.0110	0.9032 ± 0.0154	0.8988 ± 0.0635	**0.9156 ± 0.0210**
yeast4	0.2747 ± 0.1638	0.5531 ± 0.0783	0.8408 ± 0.0286	0.8028 ± 0.0249	0.8511 ± 0.0545	**0.8705 ± 0.0273**
yeast5	0.5343 ± 0.0830	0.7768 ± 0.0689	0.9635 ± 0.0032	0.9694 ± 0.0065	0.9667 ± 0.0135	**0.9835 ± 0.0037**
yeast6	0.5389 ± 0.1672	0.6401 ± 0.1491	**0.9165 ± 0.0645**	0.9076 ± 0.0619	0.8972 ± 0.0704	0.9118 ± 0.0657
zoo3	0.3896 ± 0.3335	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
Average	0.6610 ± 0.0955	0.7531 ± 0.0963	0.8908 ± 0.0381	0.8752 ± 0.0367	0.8740 ± 0.0356	**0.8948 ± 0.0413**

Table 3
Performance results on 11 multi-class imbalance datasets.

Datasets	G-mean					
	Softmax	ELM	WELM W1	WELM W2	BWELM	GELM
Balance	0.0806 ± 0.1802	0.5840 ± 0.1208	0.7810 ± 0.0380	0.7637 ± 0.0274	**0.7948 ± 0.0196**	0.7844 ± 0.0546
Contraceptive	0.4665 ± 0.0288	0.4930 ± 0.0127	0.5237 ± 0.0162	0.5102 ± 0.0201	0.5315 ± 0.0288	**0.5393 ± 0.0189**
Newthyroid	0.9487 ± 0.0445	0.8507 ± 0.0628	0.9211 ± 0.0195	0.9471 ± 0.0177	0.9511 ± 0.0141	**0.9643 ± 0.0342**
Wine	0.9691 ± 0.0228	0.9591 ± 0.0102	0.9831 ± 0.0238	0.9805 ± 0.0137	0.9820 ± 0.0129	**1.0000 ± 0.0000**
Dermatology	0.9683 ± 0.0175	0.9570 ± 0.0145	0.9610 ± 0.0195	0.9634 ± 0.0107	0.9692 ± 0.0060	**0.9815 ± 0.0315**
Yeast	0.0000 ± 0.0000	0.0000 ± 0.0000	0.2610 ± 0.0079	0.3472 ± 0.0120	0.3710 ± 0.0112	**0.3727 ± 0.0112**
Pageblocks	0.5613 ± 0.2137	0.2980 ± 0.3952	0.5308 ± 0.1374	0.5411 ± 0.0881	0.5515 ± 0.0649	**0.7936 ± 0.0171**
Penbased	0.9110 ± 0.0220	0.9602 ± 0.0096	0.9621 ± 0.0075	0.9623 ± 0.1926	**0.9631 ± 0.0080**	0.9597 ± 0.0505
Ecoli	0.3403 ± 0.4666	0.3250 ± 0.3841	0.3519 ± 0.0991	0.3382 ± 0.0037	0.3879 ± 0.0779	**0.4621 ± 0.2703**
Thyroid	0.5072 ± 0.3178	0.4973 ± 0.1563	0.5104 ± 0.0475	0.5092 ± 0.0549	0.5480 ± 0.0913	**0.5664 ± 0.0801**
Glass	0.2604 ± 0.3567	0.1170 ± 0.3122	0.2481 ± 0.2374	0.2709 ± 0.0744	**0.3108 ± 0.1009**	0.2264 ± 0.2554
Average	0.5417 ± 0.1518	0.5492 ± 0.1344	0.6394 ± 0.0594	0.6485 ± 0.0089	0.6691 ± 0.0395	**0.6955 ± 0.0749**

the other state-of-the-art methods, the evaluation metric G-mean can even improve 30% comparing with the classical ELM classifier. The average performance metrics also indicate the significant improvement of our algorithm. The advantages of our algorithm also can be shown by the average ranks of the algorithms given in Table 7.

4.3.1. *Comparing with Softmax classifier*

As can be seen in Table 2 and Table 3, GELM can achieve satisfactory performance comparing with Softmax classifier in all datasets on binary problem, in 10 datasets (90.90%) except for *glass* on multi-class problem. Specifically, on the 9 datasets (17.24%) of *glass2*, *winequalityred3vs5*, *winequalityred4*, *winequalityred8vs67*, *yeast1289vs7*, *yeast1458vs7*, *yeast4*, *zoo3* and *yeast1vs7*, GELM can achieve a improvement of more than 50% on G-mean.

There are 3 datasets where our method and Softmax classifier can both achieve perfect 100% G-mean values. The Softmax classifier can perform much better than other algorithms except for our GELM on the datasets of *pageblocks*, which also indicates that WELM W1, WELM W2, and BWELM are not optimal.

4.3.2. *Comparing with standard ELM*

As shown in Table 2 and Table 3, GELM can achieve satisfactory performance comparing with standard ELM in 54 datasets (93.10%) except for *glass0*, *glass5*, *page-blocks13vs4*, *page-blocks0* on binary problem, in 10 datasets (90.90%) except for*penbased* on multi-class problem.

Especially on the datasets of *poker89vs5*, *poker8vs6*, *winequalityred4*, *winequalityred8vs67* and *yeast1458vs7*, GELM can obtain a improvement of more than 35% on G-mean.

There are 6 datasets where both classifiers achieve perfect 100% G-mean values.

4.3.3. *Comparing with experienced weighed ELM*

Based on the results in Table 2 and Table 3, it can be seen that, GELM can achieve superior performance comparing with WELM W1 in 42 datasets (72.41%) on binary problem, in 9 datasets (81.81%) on multi-class problem. Also GELM can perform much better comparing with WELM W2 in 48 datasets (82.76%) on binary problem, in 9 datasets (81.81%) on multi-class problem. Especially on the datasets *yeast0359vs78*, *pageblocks*, and *ecoli*, GELM can gain an improvement of more than 10% on G-mean.

4.3.4. *Comparing with boosting weighed ELM*

Form Table 2 and Table 3, it can be seen that, GELM can achieve satisfactory performance in 45 datasets (77.58%) versus BWELM on binary problem, except for the 13 datasets such as *ecoli01vs235*, *ecoli0267vs35* and *glass5* etc., in 8 datasets (72.72%) except for *penbased*, *balance* and *glass* on multi-class problem. There are 7 datasets where both GELM and BWELM achieve perfect 100% G-mean values.

4.4. *Comparison results with other state-of-the-art methods*

We also compare GELM with other state-of-the-art methods using experimental results of Yu et al. [13]. The comparison results on 27 binary problems and 9 multi-class problems show in Table 4 and Table 5.

As shown in Table 4 and Table 5, GELM can achieve much better performance than other state-of-the-art methods in 21 datasets (63.63%) on binary problem and in 5 datasets (55.55%) on multi-class problem. The advantages of GELM also can be shown by the average G-mean values of the algorithms in Table 4 and Table 5 and the average ranks of the algorithms given in Table 8.

Table 4
Performance results of 27 binary problems in comparison with other state-of-the-art methods.

Datasets	G-mean				
	SMOTE-ELM	RWOS-ELM	SVM-OTHR	ODOC-ELM	GELM
ecoli3	0.8656 ± 0.0139	0.8662 ± 0.0212	0.8512 ± 0.0322	0.8696 ± 0.0250	**0.9261 ± 0.0407**
glass1	0.6832 ± 0.0262	0.6907 ± 0.0192	0.7376 ± 0.0139	0.6939 ± 0.0188	**0.7498 ± 0.0552**
haberman	0.6422 ± 0.0214	0.6475 ± 0.0217	0.6064 ± 0.0269	**0.6527 ± 0.0264**	0.6488 ± 0.0511
newthyroid1	0.9701 ± 0.0172	0.9689 ± 0.0142	0.9627 ± 0.0185	0.9808 ± 0.0076	**1.0000 ± 0.0000**
pima	0.7512 ± 0.0035	0.7505 ± 0.0133	0.6781 ± 0.0137	0.7534 ± 0.0116	**0.7609 ± 0.0350**
vehicle1	0.8386 ± 0.0089	0.8276 ± 0.0107	0.7865 ± 0.0170	**0.8437 ± 0.0092**	0.8382 ± 0.0354
wisconsin	0.9646 ± 0.0052	0.9702 ± 0.0031	0.9660 ± 0.0036	0.9744 ± 0.0036	**0.9802 ± 0.0059**
yeast3	0.9176 ± 0.0056	0.9105 ± 0.0077	0.8917 ± 0.0079	**0.9219 ± 0.0069**	0.9156 ± 0.0210
ecoli4	0.9179 ± 0.0218	0.9207 ± 0.0275	0.8671 ± 0.0734	0.9343 ± 0.0290	**0.9872 ± 0.0133**
shuttlec2vsc4	0.9935 ± 0.0029	0.9926 ± 0.0031	0.9960 ± 0.0007	0.9933 ± 0.0026	**1.0000 ± 0.0000**
vowel0	0.9873 ± 0.0039	0.9896 ± 0.0021	**1.0000 ± 0.0000**	0.9811 ± 0.0094	0.9493 ± 0.0494
yeast4	0.7999 ± 0.0211	0.8017 ± 0.0085	0.7595 ± 0.0258	0.8109 ± 0.0097	**0.8705 ± 0.0273**
yeast5	0.9498 ± 0.0122	0.9544 ± 0.0128	0.8987 ± 0.0237	0.9588 ± 0.0143	**0.9835 ± 0.0037**
yeast1vs7	0.7379 ± 0.0510	0.7322 ± 0.0462	0.6127 ± 0.0615	0.7354 ± 0.0617	**0.7837 ± 0.0634**
ecoli01vs235	0.8704 ± 0.0257	0.8700 ± 0.0222	0.7969 ± 0.0552	0.8965 ± 0.0271	**0.9397 ± 0.0648**
ecoli01vs5	0.8932 ± 0.0602	0.8854 ± 0.0548	0.8943 ± 0.0210	0.9040 ± 0.0304	**0.9448 ± 0.0591**
ecoli034vs5	0.8701 ± 0.0574	0.8669 ± 0.0642	0.8688 ± 0.0660	0.8774 ± 0.0754	**0.9478 ± 0.0741**
ecoli067vs35	0.8475 ± 0.0857	0.8507 ± 0.0709	0.7877 ± 0.0714	0.8799 ± 0.0840	**0.9422 ± 0.0535**
ecoli067vs5	0.8447 ± 0.0539	0.8475 ± 0.0602	0.8568 ± 0.0526	0.8532 ± 0.0568	**0.9559 ± 0.0393**
led7digit0245 6789vs1	0.8892 ± 0.0280	0.8901 ± 0.0254	0.8508 ± 0.1073	0.8947 ± 0.0153	**0.9468 ± 0.0450**
yeast02579vs 368	0.8839 ± 0.0113	0.8852 ± 0.0112	0.8842 ± 0.0072	0.8970 ± 0.0113	**0.9164 ± 0.0348**
yeast0359vs78	**0.8231 ± 0.0394**	0.8176 ± 0.0328	0.6396 ± 0.0585	0.6583 ± 0.0508	0.7851 ± 0.0437
poker89vs5	0.4976 ± 0.1551	0.5000 ± 0.1052	0.5109 ± 0.0795	0.5054 ± 0.0975	**0.6519 ± 0.1244**
poker8vs6	**0.8842 ± 0.0765**	0.8133 ± 0.1024	0.6309 ± 0.0869	0.7949 ± 0.1319	0.8446 ± 0.1297
shuttle2vs5	0.9951 ± 0.0007	0.9949 ± 0.0006	**1.0000 ± 0.0000**	0.9985 ± 0.0005	**1.0000 ± 0.0000**
shuttle6vs23	0.8441 ± 0.1209	0.8501 ± 0.1134	0.8249 ± 0.1525	0.9330 ± 0.0896	**0.9977 ± 0.0051**
winequalityred4	0.6801 ± 0.0272	0.6705 ± 0.0297	0.5401 ± 0.0701	0.6726 ± 0.0186	**0.7311 ± 0.0430**
Average	0.8460 ± 0.0354	0.8432 ± 0.0334	0.8037 ± 0.0424	0.8470 ± 0.0343	**0.8888 ± 0.0414**

Table 5
Performance results of 9 multi-class problems in comparison with other state-of-the-art methods.

Datasets	G-mean				
	SMOTE-ELM	RWOS-ELM	SVM-OTHR	ODOC-ELM	GELM
Balance	0.7876 ± 0.0165	0.7989 ± 0.0165	**0.8696 ± 0.0171**	0.8268 ± 0.0238	0.7844 ± 0.0546
Contraceptive	0.5397 ± 0.0091	**0.5416 ± 0.0075**	0.5307 ± 0.0077	0.5217 ± 0.0069	0.5393 ± 0.0189
Newthyroid	0.9167 ± 0.0211	0.9155 ± 0.0196	0.9488 ± 0.0173	0.9273 ± 0.0181	**0.9643 ± 0.0342**
Wine	0.9808 ± 0.0070	0.9796 ± 0.0072	0.9798 ± 0.0052	0.9820 ± 0.0084	**1.0000 ± 0.0000**
Dermatology	0.9610 ± 0.0079	0.9588 ± 0.0062	0.9570 ± 0.0082	0.9626 ± 0.0096	**0.9815 ± 0.0315**
Yeast	0.3138 ± 0.1025	0.3249 ± 0.0858	0.4484 ± 0.0778	**0.4931 ± 0.1153**	0.3727 ± 0.0112
Pageblocks	0.4753 ± 0.1005	0.4667 ± 0.1132	0.4479 ± 0.0808	0.4635 ± 0.0906	**0.7936 ± 0.0171**
Penbased	0.9611 ± 0.0032	0.9638 ± 0.0026	0.9629 ± 0.0037	**0.9649 ± 0.0049**	0.9597 ± 0.0505
Ecoli	0.3542 ± 0.0698	0.3555 ± 0.0757	0.3391 ± 0.0630	0.3936 ± 0.0674	**0.4621 ± 0.2703**
Average	0.6989 ± 0.0375	0.5989 ± 0.0371	0.7204 ± 0.0312	0.7261 ± 0.0383	**0.7620 ± 0.0543**

4.5. Experimental results on Youtube-8M database

The comparable experimental results of these five methods, i.e., ELM, WELM W1, WELM W2, BWELM, GELM, under selected imbalance dataset in Youtube-8M are given in Table 6.

For performance estimation metrics, we adopted G-mean, Accuracy and F-score to compare the performance of various learning algorithms. F-score metric [35] is to transform the F-measure for a multi-class problem.

As shown in Table 6, GELM can achieve better performance than other methods on Youtube-8M database. The experimental results indicate that our GELM model is an effective and efficient algorithm for solving the class imbalance problem in varied datasets. Table 6 provides the total running time of various learning algorithms on the YouTube-8M database. From Table 6, the execution time of GELM is a little slower than WELM W1 and WELM W2. As our proposed GELM didn't employ the optimization search procedure in the training step, our algorithm is several times faster than BWELM.

4.6. Statistical significance of results

In order to verify if the results obtained by GELM are significantly superior, statistically, to the other competing methods, we employed Friedman test [36] and adopted Nemenyi post-hoc test [37] to examine whether the proposed algorithm is distinctive among a $1 \times K$ comparison.

Friedman test is a non-parametric test commonly employed for comparing multiple methods on multiple datasets. This approach uses only comparisons between the rank of a method (For a given dataset, the algorithm with the highest G-mean value is ranked as 1, the algorithm with the second highest G-mean value is ranked as 2, and so forth. In case the methods obtain same G-mean value, an average rank is applied to each method. This task is carried out for all datasets and finally an average rank is calculated.), and thus requires no assumptions about the distribution of G-mean to be made. The post-hoc procedure allows us to know whether a hypothesis of comparison of means could be rejected at a specified level of significance α.

Each dataset is treated as a separate test, giving one rank measurement per method and dataset. The analysis then consists of

Table 6
Experimental results on Youtube-8M database.

Experiment index	ELM	WELM W1	WELM W2	BWELM	GELM
G-mean	0.5884 ± 0.0385	0.6175 ± 0.0215	0.6148 ± 0.0253	0.6164 ± 0.0145	**0.6259 ± 0.0125**
Accuracy	0.6854 ± 0.0163	0.6848 ± 0.0635	0.6835 ± 0.0532	0.6842 ± 0.0567	**0.6879 ± 0.0365**
F-score	0.6284 ± 0.0524	0.6440 ± 0.0705	0.6368 ± 0.1065	0.6532 ± 0.0865	**0.6568 ± 0.0475**
Running time (s)	0.3410 ± 0.0012	0.8664 ± 0.0043	1.0704 ± 0.0093	8.0378 ± 0.0104	2.0578 ± 0.0014

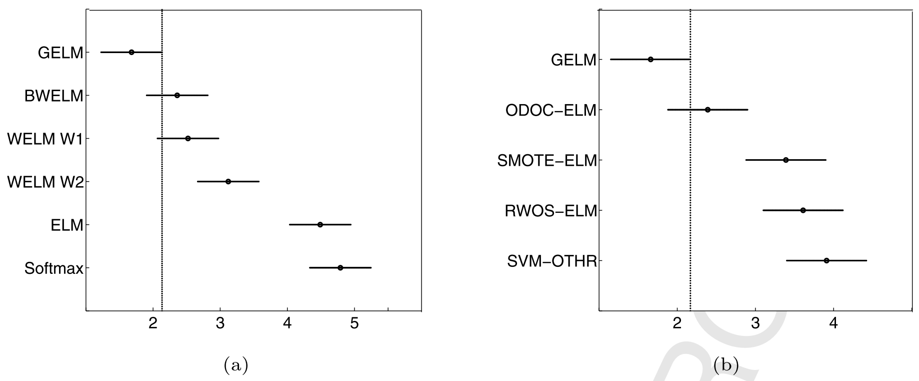

Fig. 2. Statistical analysis of comparison results on (a) standard classification datasets and (b) other state-of-the-art methods.

Table 7
The average ranks of the algorithms in Standard Classification Datasets.

Algorithms	Average rank
Softmax	4.79
ELM	4.49
WELM W1	2.52
WELM W2	3.12
BWELM	2.36
GELM	**1.68**

Table 8
The average ranks of the algorithms in comparison with other state-of-the-art methods.

Algorithms	Average rank
SMOTE-ELM	3.39
RWOS-ELM	3.61
SVM-OTHR	3.91
ODOC-ELM	2.39
GELM	**1.66**

two steps: (i) the null hypothesis is made that the methods are equivalent and therefore their ranks should be equal. The hypothesis is tested by the Friedman test, which follows a F distribution; (ii) having rejected the null hypothesis the differences in ranks are analyzed by the Nemenyi test. If the rank difference between a pair of classifiers is larger than the critical difference (CD) at a certain confidence level, the two classifiers are considered statistically different.

For a confidence level of $\alpha = 0.05$ and given the 6 methods (Softmax, ELM, WELM W1, WELM W2, BWELM, GELM) tested over standard classification datasets, CD is calculated as 0.9078. The average ranks of 6 algorithms are shown in Table 7 and the statistical results are provided in Fig. 2(a).

Fig. 2 visualizes the results of the statistical analysis using the Friedman test picture [37]. The y-axis indicates the algorithms and x-axis shows the average ranks over datasets for each algorithm. The average ranks of the algorithms are shown in ascending rank order on the x-axis. The circle points in each algorithm represent the average ranks of the algorithms and the horizontal line segments that consider the circle points as its center points shows the sizes of critical difference (CD). As can be observed from Fig. 2, if a pair of the algorithms are not statistically different, there is substantial overlap between the line segments of the algorithms and otherwise.

From Table 7, we observe that GELM has obtained the lowest rank, thus it is the best algorithm. In addition, as can be seen in Fig. 2(a), all average rank difference except for BWELM and WELM W1 are larger than CD, meaning the null-hypothesis of equality can be rejected in these cases.

We also performed the statistical analysis on comparison results with other state-of-the-art methods (SMOTE-ELM, RWOS-ELM, SVM-OTHR and ODOC-ELM). The average ranks of 5 algorithms (SMOTE-ELM, RWOS-ELM, SVM-OTHR, ODOC-ELM and GELM) are shown in Table 8. In this case, CD = 1.0167 at $\alpha = 0.05$. The statistical results are provided in Fig. 2(b).

While for the average rank difference, we can state that GELM is significant better than SMOTE-ELM, RWOS-ELM and SVM-OTHR at $\alpha = 0.05$, but we cannot say that GELM is significant different from ODOC-ELM, although it has obviously lower average ranks than ODOC-ELM.

5. Conclusions

In this paper, a cost function of ELM is newly presented on G-mean widely used as evaluation metric in the imbalance data learning and GELM is proposed according as this cost function.

The effectiveness of GELM is proven by experiments conducted using 58 binary datasets and 11 multi-class datasets which have different degrees of imbalance, as well as one large-scale video dataset from YouTube-8M. For performance estimation metrics on YouTube-8M, we adopted G-mean, accuracy and F-score to compare the performance of various learning algorithms. We performed experiments comparing other computational methods, and also provided the statistical results to further estimate the performance of these learning algorithms. Experimental results indicate that GELM can be superior to many popular and state-of-the-art algorithms. The future work will focus on improving more effectively the cost function of ELM in order to overcome the class imbalance problem, and working the efficient learning algorithm according as the distribution property of an imbalance dataset.

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] G.-B. Huang, Q.-Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[2] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern., Part B, Cybern. 42 (2) (2012) 513–529.
[3] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (16–18) (2007) 3056–3062.
[4] G.-B. Huang, D. Wang, Y. Lan, Extreme learning machines: a survey, Int. J. Mach. Learn. Cybern. 2 (2) (2011) 107–122.
[5] M. Xia, Y. Zhang, L. Weng, X. Ye, Fashion retailing forecasting based on extreme learning machine with adaptive metrics of inputs, Knowl.-Based Syst. 36 (2012) 253–259.
[6] F.L. Chen, T.Y. Ou, Sales forecasting system based on gray extreme learning machine with Taguchi method in retail industry, Expert Syst. Appl. 38 (3) (2011) 1336–1345.
[7] A. Samat, P. Du, S. Liu, J. Li, L. Cheng, E^2LMs: ensemble extreme learning machines for hyperspectral image classification, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 7 (4) (2014) 1060–1069.
[8] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, Smote: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (1) (2002) 321–357.
[9] H. Zhang, M. Li, RWO-sampling: a random walk over-sampling approach to imbalanced data classification, Inf. Fusion 20 (2014) 99–116.
[10] F. Charte, A.J. Rivera, M.J. del Jesus, F. Herrera, MLSMOTE: approaching imbalanced multilabel learning through synthetic instance generation, Knowl.-Based Syst. 89 (2015) 385–397.
[11] V. López, S. del Río, J.M. Benítez, F. Herrera, Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data, Fuzzy Sets Syst. 258 (1) (2015) 5–38.
[12] Z.H. Zhou, X.Y. Liu, On multi-class cost-sensitive learning, Comput. Intell. 26 (2010) 232–257.
[13] H. Yu, C. Sun, X. Yang, W. Yang, J. Shen, Y. Qi, ODOC-ELM: optimal decision outputs compensation-based extreme learning machine for classifying imbalanced data, Knowl.-Based Syst. 92 (15) (2016) 55–70.
[14] H. Yu, C. Mu, C. Sun, W. Yang, X. Yang, X. Zuo, Support vector machine-based optimized decision threshold adjustment strategy for classifying imbalanced data, Knowl.-Based Syst. 76 (2015) 67–78.
[15] B. Krawczyk, M. Woźniak, G. Schaefer, Cost-sensitive decision tree ensembles for effective imbalanced classification, Appl. Soft Comput. 14 (Part C) (2014) 554–562.
[16] Q. Li, Y. Mao, A review of boosting methods for imbalanced data classification, PAA Pattern Anal. Appl. 17 (2014) 679–697.
[17] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, in: Neurocomputing: Foundations of Research, MIT Press, Cambridge, MA, USA, 1988, pp. 696–699.
[18] M.T. Hagan, M.B. Menhaj, Training feedforward networks with the Marquardt algorithm, IEEE Trans. Neural Netw. 5 (6) (1994) 989–993.
[19] W. Zong, G.-B. Huang, Y. Chen, Weighted extreme learning machine for imbalance learning, Neurocomputing 101 (3) (2013) 229–242.
[20] K. Li, X. Kong, Z. Lu, L. Wenyin, J. Yin, Boosting weighted ELM for imbalanced learning, Neurocomputing 128 (5) (2014) 15–21.
[21] R.E. Schapire, The boosting approach to machine learning: an overview, in: Nonlinear Estimation and Classification, Springer, 2003, pp. 149–171.
[22] C.R. Rao, S.K. Mitra, Generalized Inverse of Matrices and Its Applications, Wiley, New York, 1971.
[23] D. Serre, Matrices: Theory and Applications, Springer-Verlag, New York, 2002.
[24] R. Fletcher, Practical Methods of Optimization, Constrained Optimization, vol. 2, John Wiley & Sons, New York, 1981.
[25] N.N. Schraudolph, J. Yu, S. Günter, A stochastic quasi-Newton method for online convex optimization, in: Proc. of 11th International Conference on Artificial Intelligence and Statistics, Vilamoura, Algarve, Portugal, 2007, pp. 436–443.
[26] A. Mokhtari, A. Ribeiro, RES: regularized stochastic BFGS algorithm, IEEE Trans. Signal Process. 62 (23) (2014) 6089–6104.
[27] J.L. Morales, J. Nocedal, Remark on "algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization", ACM Trans. Math. Softw. 38 (1) (2011) 7.
[28] R.H. Byrd, P. Lu, J. Nocedal, A limited-memory algorithm for bound-constrained optimization, SIAM J. Sci. Comput. 16 (5) (1995) 1190–1208.
[29] C. Zhu, R.H. Byrd, P. Lu, J. Nocedal, Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization, ACM Trans. Math. Softw. 23 (4) (1997) 550–560.
[30] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Mult.-Valued Log. Soft Comput. 17 (2–3) (2011) 255–287.
[31] S. Abu-El-Haija, N. Kothari, J. Lee, A. Natsev, G. Toderici, B. Varadarajan, S. Vijayanarasimhan, Youtube-8m: a large-scale video classification benchmark, CoRR, arXiv:1609.08675.
[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, F.-F. Li, ImageNet: a large-scale hierarchical image database, IEEE Conf. Comput. Vis. Pattern Recognit. (2009) 248–255.
[33] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, CoRR, arXiv:1409.1556.
[34] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[35] A. Ozgur, L. Ozgur, T. Gungor, Text categorization with class-based and corpus-based keyword selection, in: Lecture Notes Comput. Sci., vol. 3733, 2005, pp. 606–615.
[36] S. Garcia, A. Fernandez, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Comput. (2009) 959–977.
[37] Z.-H. Zhou, Machine Learning, Qinghua University Press, Beijing, 2016.

Jong Hyok Ri received the B.S. and M.S. degrees in information mathematics and computer Science from Kim Il Sung University, DPR Korea, in 2003 and 2007, respectively. He is currently with Institute of Information Technology, Kim Il Sung University, DPR Korea. His research interests include pattern recognition, machine learning and data mining.

Hun Kim received the M.S. and Ph.D. degrees in computer Science from Kim Il Sung University, DPR Korea. He is currently a professor in the School of Information Science, Kim Il Sung University, DPR Korea. His research interests include pattern recognition and computer network.