IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 12, 2021, accepted January 8, 2022, date of publication January 13, 2022, date of current version January 24, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3142724

An Improved Extreme Learning Machine for Imbalanced Data Classification

XIAOPENG ZHANG AND LIANGXI QIN
School of Computer, Electronics and Information, Guangxi University, Nanning 530004, China
Guangxi Key Laboratory of Multimedia Communications and Network Technology, Nanning 530004, China

Corresponding author: Liangxi Qin (qin_lx@126.com)

This work was supported in part by the Guangxi Key Research and Development Project of Guangxi, China, under Grant Guike AB16380260, and in part by the National Natural Science Foundation of China under Grant 62162003.

ABSTRACT In the field of machine learning, Extreme Learning Machine (ELM) has been widely used in classification and regression tasks. However, like many traditional machine learning algorithms, the classification results of ELM are often not good enough when facing imbalanced data. For this reason, we proposed an extreme learning machine algorithm with output weight adjustment called OWA-ELM, which can make the decision boundary of ELM move to majority classes, and improve the classification performance of imbalanced data. Specifically, in ELM, we add a reasonable increment Δ to the connection weights between hidden layer neurons and minority output neurons, so that the output value of minority output neuron increases. Finally, the classification accuracy of the minority class samples can be improved without significantly affecting the classification of the majority class samples. The performance of OWA-ELM was compared with ELM, WELM, CS-ELM and CCR-ELM. In the experiments on 22 data sets, the OWA-ELM algorithm has achieved 9 times optimal and 4 times suboptimal results on G-mean. In F-measure, 13 times optimal and one suboptimal results were obtained. Therefore, the OWA-ELM algorithm is effective to deal with imbalanced data classification.

INDEX TERMS Extreme learning machine, imbalanced data, classification, output weights adjustment.

I. INTRODUCTION

In the practical classification applications of machine learning, many data sets are often imbalanced [1]. The classification problem of imbalanced data is particularly prominent in many fields such as computer vision [2], [3], medical science [4], information security [5] and industry [6]. When the traditional classification methods deal with such data sets, the classification results tend to be biased towards the majority class, and it is easy to ignore the minority class [7]. This is because these classifiers generally assume that the distribution of samples is balanced [8].

Extreme Learning Machine (ELM) is a single hidden layer feedforward neural network [9], [10]. The input weights and the threshold of hidden layer neurons are randomly generated, and there is no need to adjust during the learning process. Even, it does not use the Backpropagation algorithm which is high time complexity to obtain the only optimal solution. Therefore, ELM has the advantages of fast learning speed and good generalization performance. It has been widely used in data classification problems in various fields, such as face recognition [11], [12], military [13], image processing [14], [15] and medical diagnosis [16], [17] and etc. However, just like traditional machine learning algorithms, ELM was not designed to classify imbalanced data, which leads to its poor classification effect on imbalanced data [18]. When ELM is faced with imbalanced data sets, the classification results tend to favor the majority classes, and it is easy to ignore the minority classes.

In order to overcome the shortcomings of ELM in classifying imbalanced data, Zong and Huang *et al.* [19] proposed a weighted extreme learning machine (WELM) algorithm based on the idea of cost sensitivity. Before the training process of WELM, the samples are given weights, and the weight of the minority samples is greater than the weight of the majority samples, so as to improve the classification accuracy of the minority samples. In practical applications, two weighting schemes can be selected according to the sample distribution. Yu *et al.* [20] proposed the label-weighted extreme learning machine (LW-ELM) algorithm to improve WELM. LW-ELM has a faster training speed on large-scale data sets by eliminating a large-matrix multiplication operation. At the same time, the authors also provided their two weighting schemes. Xiao *et al.* [21]

The associate editor coordinating the review of this manuscript and approving it for publication was Thomas Canhao Xu.

proposed a class-specific cost regulation extreme learning machine (CCR-ELM). CCR-ELM improves the classification accuracy of imbalanced data by modifying the optimization function of the ELM algorithm and introducing regularization parameters for different classes as a balance between structural risk and empirical risk. In order to overcome the shortcomings of CCR-ELM and reduce the parameters, Raghuwanshi *et al.* [22] proposed the CS-ELM algorithm. Compared with CCR-ELM, the parameters of CS-ELM are reduced by one, and the problem of degradation to ELM caused by different orders of magnitude of two regularization parameters in CCR-ELM is avoided. In order to solve the problem of data imbalance in online learning, Mao *et al.* [23] proposed a new online sequential extreme learning machine algorithm which with two-stage hybrid strategy. In the offline phase, principal curve and database technique are used to model the data. In the online phase, they proposed a leave-one-out cross-validation method using Sherman–Morrison matrix inversion to solve the online imbalanced data. Meanwhile, add-delete mechanism is used to update the network weights.

Considering the shortcomings of ELM algorithm and the property of imbalanced data, an improved extreme learning machine with output weight adjustment (OWA-ELM) was proposed. The proposed algorithm improves the classification accuracy of minority classes by adding a reasonable increment Δ to the connection weights between hidden layer neurons and minority output neuron in ELM.

The reminder of this paper is organized as follows. Section II introduces related algorithms. The proposed method is presented in Section III. Section IV provides the experimental results and analysis. Finally, the conclusion is given in Section V.

II. RELATED WORK

In this section, the ELM, WELM and CCR-ELM will be briefly introduced.

A. EXTREME LEARNING MACHINE

Extreme learning machine (ELM) [9], [10] is a single hidden layer feedforward neural network. Its structure is shown in figure 1. As mentioned above, in ELM, the input weights and the bias of the hidden layer neurons are randomly generated and not need to be changed during the training process. The output weight matric can be directly calculated and not need to be adjusted by the backpropagation algorithm. Therefore, ELM has the advantages of fast training speed and good generalization performance.

Let $[\boldsymbol{x}_i, \boldsymbol{t}_i](i \in 1, 2, \ldots, N)$ represents N training samples. More precisely, the true label matrix of the i-th training sample $\boldsymbol{x}_i = [x_{i1}, x_{i2} \ldots, x_{in}]^T \in R^n$ is $\boldsymbol{t}i = [t_{i1}, t_{i2} \ldots, t_{im}]^T \in R^m$, where the superscript T represents the transposition of a matrix or a vector. Let n represents the number of neurons in the input layer, which is also the number of features of the samples. Let m represents the number of neurons in the output layer, which is also the class number of

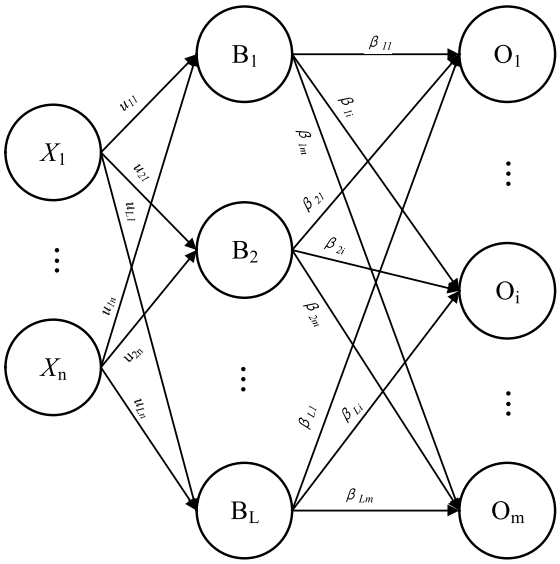

FIGURE 1. ELM neural network structure.

data set. Let L denote the number of neurons in the hidden layer. Then the input weights can be expressed as $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2 \ldots, \boldsymbol{u}_L]^T \in R^{L\times n}$, where, $\boldsymbol{u}_i = [u_{i1}, u_{i2} \ldots, u_{in}] \in R^n, i \in 1, 2, \ldots, L$. And the bias vector of neurons in the hidden layer can be expressed as $\boldsymbol{b} = [b_1, b_2 \ldots, b_L]^T \in R^L$. Then the output of the i-th training sample in the hidden layer is as follows:

$$\boldsymbol{h}(\boldsymbol{x}_i) = g(\boldsymbol{U}\boldsymbol{x}_i + \boldsymbol{b}) \tag{1}$$

Here, $g(.)$ is the activation function, and the output matrix of hidden layer composed of all training samples can be denoted as $\boldsymbol{H}$ which is with the dimension of $n \times L$:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{h}(\boldsymbol{x}_N) \end{bmatrix} = \begin{bmatrix} g(\boldsymbol{u}_1\boldsymbol{x}_1 + b_1) & \cdots & g(\boldsymbol{u}_L\boldsymbol{x}_1 + b_L) \\ \vdots & & \vdots \\ g(\boldsymbol{u}_1\boldsymbol{x}_N + b_1) & \cdots & g(\boldsymbol{u}_L\boldsymbol{x}_N + b_L) \end{bmatrix} \tag{2}$$

Let β_{ij} denote the connection weights between the hidden layer neuron $\mathrm{B_i}$ and the output neuron $\mathrm{O_j}$, then the output weight matrix can be expressed as follows:

$$\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_m] = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1m} \\ \vdots & & \vdots \\ \beta_{L1} & \cdots & \beta_{Lm} \end{bmatrix} \in R^{L\times m} \tag{3}$$

The optimization function of ELM is as follows:

$$\begin{aligned} &\text{Minimize}: \quad \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum\nolimits_{i=1}^{N}\|\xi i\|^2 \\ &\text{subject to}: \ \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} = \boldsymbol{t}_i^T - \boldsymbol{\xi}_i^T, \quad i = 1, 2, \ldots, N \end{aligned} \tag{4}$$

Here $\boldsymbol{\xi}_i = [\xi_{i1}, \xi_{i2} \ldots, \xi_{in}]^T$ is a vector composed of training errors of sample $\boldsymbol{x_i}$ at m output nodes. According to the KKT

theorem, the solution of (4) is as follows:

$$\boldsymbol{\beta} = \begin{cases} \boldsymbol{H}^T \left(\dfrac{\boldsymbol{I}}{C} + \boldsymbol{H}\boldsymbol{H}^T \right)^{-1} \boldsymbol{T}, & \boldsymbol{N} < \boldsymbol{L} \\ \left(\dfrac{\boldsymbol{I}}{C} + \boldsymbol{H}^T\boldsymbol{H} \right)^{-1} \boldsymbol{H}^T\boldsymbol{T}, & \boldsymbol{N} > \boldsymbol{L} \end{cases} \tag{5}$$

where $\boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{bmatrix} = \begin{bmatrix} t_{11} & \cdots & t_{1m} \\ \vdots & & \vdots \\ t_{N1} & \cdots & t_{Nm} \end{bmatrix} \in R^{N\times m}$, $\boldsymbol{I}$ is a unit matrix with appropriate dimensions. For any given test sample $\boldsymbol{x}$, the final prediction output is:

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{cases} sign\, \boldsymbol{H}^T \left(\dfrac{\boldsymbol{I}}{C} + \boldsymbol{H}\boldsymbol{H}^T \right)^{-1} \boldsymbol{T}, & \boldsymbol{N} < \boldsymbol{L} \\ sign \left(\dfrac{\boldsymbol{I}}{C} + \boldsymbol{H}^T\boldsymbol{H} \right)^{-1} \boldsymbol{H}^T\boldsymbol{T}, & \boldsymbol{N} > \boldsymbol{L} \end{cases} \tag{6}$$

Here, $\boldsymbol{f}(\boldsymbol{x}) = [\boldsymbol{f}_1(\boldsymbol{x}), \ldots, \boldsymbol{f}_m(\boldsymbol{x})]$, which means the output matrix. Then the predicted label of sample $\boldsymbol{x}$ is:

$$\text{label}(\boldsymbol{x}) = \underset{i}{\text{argmax}}\, \boldsymbol{f}_i(\boldsymbol{x}), \quad i = 1, 2, \ldots, m \tag{7}$$

B. WEIGHTED EXTREME LEARNING MACHINE

Weighted extreme learning machine (WELM) [19] is a cost-sensitive ELM algorithm proposed in 2013 by Zong and Huang *et al.* that incorporated cost-sensitive ideas into the ELM algorithm, which can better classify imbalanced data.

In the WELM algorithm, a training sample weight matrix $\boldsymbol{W} = diagonal\{W_{\text{ii}}\}(i = 1,2,\ldots, N)$ with a dimension of $N \times N$ is introduced. Each training sample is assigned a weight. In the diagonal matrix $\boldsymbol{W}$, *Wii* corresponds to the weight assigned to the sample $\boldsymbol{x_i}$. The weight of minority class samples is greater than the weight of majority class samples, which makes the classifier pay more attention to the minority class samples. The authors provide two sample weighting schemes:

The first weighting scheme W1:

$$W_{ii} = 1/\#(t_i) \tag{8}$$

The second weighting scheme W2:

$$W_{ii} = \begin{cases} 0.618/\#(t_i), & ti > AVG(t_i) \\ 1/\#(ti), & t_i \le AVG(t_i) \end{cases} \tag{9}$$

where $\#(t_i)$ represents the number of instances of the i-th class, $i = 1, \ldots,$ m. AVG(t_i) is the average number of samples of each class in the training set.

The optimization function of WELM is as follows:

$$\begin{aligned} \text{Minimize}: \quad & \frac{1}{2}\|\beta\|^2 + \frac{1}{2}C\boldsymbol{W}\sum\nolimits_{i=1}^{N} \|\xi_i\|^2 \\ \text{subject to}: \quad & \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} = \boldsymbol{t}_i^T - \boldsymbol{\xi}_i^T, \quad i = 1, 2, \ldots, N \end{aligned} \tag{10}$$

According to the KKT theorem, the solution of (10) is as follows:

$$\boldsymbol{\beta} = \begin{cases} \boldsymbol{H}^T \left(\dfrac{\boldsymbol{I}}{C} + \boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T \right)^{-1} \boldsymbol{W}\boldsymbol{T}, & \boldsymbol{N} < \boldsymbol{L} \\ \left(\dfrac{\boldsymbol{I}}{C} + \boldsymbol{H}^T\boldsymbol{W}\boldsymbol{H} \right)^{-1} \boldsymbol{H}^T\boldsymbol{W}\boldsymbol{T}, & \boldsymbol{N} > \boldsymbol{L} \end{cases} \tag{11}$$

C. CLASS-SPECIFIC COST REGULATION EXTREME LEARNING MACHINE

Class-specific cost regulation extreme learning machine (CCR-ELM) [21] was proposed by Xiao *et al.* in 2017, which aims that the ELM can classify imbalanced data better. Its biggest feature is that it introduces regularization parameters for different types of loss costs, which can be known through its optimization function:

$$\begin{aligned} \text{Minimize}: \quad & \frac{1}{2}\|\beta\|^2 + \frac{1}{2}C^+ \sum\nolimits_{i=1|ti=+1}^{N+} \|\xi i\|^2 \\ & + \frac{1}{2}C^- \sum\nolimits_{i=1|ti=-1}^{N-} \|\xi i\|^2 \\ \text{subject to}: \quad & \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta} = \boldsymbol{t}_i^T - \boldsymbol{\xi}_i^T, \quad i = 1, 2, \ldots, N \end{aligned} \tag{12}$$

In (12), C^+ and C^- are the class-specific cost regulation parameters. N^+ and N^- represent the number of minority samples and majority samples, respectively. The solution of the above formula is

$$\boldsymbol{\beta} = \begin{cases} \boldsymbol{H}^T \left(\dfrac{\boldsymbol{I}}{C^+} + \dfrac{\boldsymbol{I}}{C^-} + \boldsymbol{H}\boldsymbol{H}^T \right)^{-1} \boldsymbol{T}, & \boldsymbol{N} < \boldsymbol{L} \\ \left(\dfrac{\boldsymbol{I}}{C^+} + \dfrac{\boldsymbol{I}}{C^-} + \boldsymbol{H}^T\boldsymbol{H} \right)^{-1} \boldsymbol{H}^T\boldsymbol{T}, & \boldsymbol{N} > \boldsymbol{L} \end{cases} \tag{13}$$

III. PROPOSED METHOD

For balanced data, the output weight matrix $\boldsymbol{\beta}$ obtained by (5) is the optimal solution. But for imbalanced data, the $\boldsymbol{\beta}$ may not optimal. So, it tends to ignore the minority classes when ELM is used in classifying imbalanced data. In ELM algorithm, $\|\beta\|$ is required as small as possible to avoid the structural risk of the model and resulting in weak generalization ability. However, if only slightly increase each element in the column vector that determines the output value of the minority output neuron in the output weight matrix $\boldsymbol{\beta}$, it may not cause too much structural risk, but helps to improve the classification accuracy of minority class samples. For this reason, an Extreme learning machine with output weight adjustment (OWA-ELM) was proposed in this paper. By slightly increasing the corresponding column in the output weight $\boldsymbol{\beta}$, that is, slightly increasing the weights between hidden layer neurons and the minority output neuron, the output of the minority output neuron is increased. Under the condition that does not significantly affect the correct classification of samples of the majority class, when increasing the value of minority output neurons, the probability of the samples of the minority class being correctly classified increases.

In figure 1, for any sample $\boldsymbol{x}$, the output of neuron O_i is calculated as follows:

$$f_i(\boldsymbol{x}) = H_1\beta_{1i} + H_2\beta_{2i} + \ldots + H_L\beta L_i \tag{14}$$

Here, H_l represents the output of the hidden layer neuron B_l, $l = 1, \ldots, L$. Assuming that the output neuron O_i corresponds to the minority class. Then, according to (7), for a sample $\boldsymbol{x}$, when the output value of O_i is greater than other output neurons, $\boldsymbol{x}$ is predicted to be a minority class. In ELM, in order to pursue higher accuracy, the output value of the minority sample $\boldsymbol{x}$ on O_i is often slightly smaller than the output value of some of other output neurons, that is, the minority sample is mistakenly classified as the majority class. According to (14), for a minority sample $\boldsymbol{x}$, when the output of the hidden layer neuron $H_1, H_2, \ldots, H_L$ is constant, in order to slightly increase its output on O_i, one possible way is to slightly increase the weights $\beta_{1i}, \beta_{2i}, \ldots, \beta_{Li}$, which are the connections between the hidden neurons $B_1, B_2, \ldots, B_L$ and O_i. Since proposed method adds the same number Δ to $\beta_{1i}, \beta_{2i}, \ldots, \beta_{Li}$, according to the matrix representation of output weight $\boldsymbol{\beta}$ of ELM in (2), the new output weight $\boldsymbol{a\_\beta}$ after fine-tuning only needs to add Δ to each element of $\boldsymbol{\beta_i}$:

$$\begin{aligned}\boldsymbol{\alpha\_\beta} &= [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_i + \boldsymbol{\delta}, \ldots, \boldsymbol{\beta}_m] \\ &= \begin{bmatrix} \beta_{11} & \cdots & \beta_{1i+\Delta} & \cdots & \beta_{1m} \\ \vdots & & \vdots & & \vdots \\ \beta_{L1} & \cdots & \beta_{Li+\Delta} & \cdots & \beta_{Lm} \end{bmatrix} \in R^{L\times m}\end{aligned} \tag{15}$$

Here, $\delta = [\Delta, \Delta, \ldots, \Delta]^T \in R^L$ ∘

The training process of OWA-ELM algorithm is described as Algorithm 1.

Algorithm 1 OWA-ELM

Input: training set: $\{\boldsymbol{x_i}, \boldsymbol{t_i} | \boldsymbol{x_i} \in R^n, \quad \boldsymbol{t_i} \in R^m, i = 2, \ldots, N\}$
Number of hidden neurons: L
Regularization parameter: C
Activation function: g(.)
Increment value: Δ
Output: Output weight $\boldsymbol{a\_\beta}$
1. Randomly generate input weight $\boldsymbol{U}$ and hidden layer neuron bias $\boldsymbol{b}$;
2. Calculate the hidden layer output matrix: $\boldsymbol{H} = g(\boldsymbol{x}\times\boldsymbol{U} + \boldsymbol{b})$;
3. Calculate the output weight of ELM $\boldsymbol{\beta}$ using (5);
4. Use (15) to adjust $\boldsymbol{\beta}$ obtained in step 3 to obtain the adjusted output weight $\boldsymbol{a\_\beta}$;
Return $\boldsymbol{a\_\beta}$;

Because the proposed method only adds a tiny value (Δ) to each element in one of the columns of $\boldsymbol{\beta}$, the structural risk caused by the change of $\boldsymbol{\beta}$ may be little. Generally, Δ is much smaller than the number to be added, which will not cause too much negative impact on the neural network. Subsequent experiments will also prove that the OWA-ELM algorithm does not have structural risks such as over-fitting, but has achieved good results in the evaluation of G-mean and F-measure.

TABLE 1. Datast details.

Dataset	#Instances	#Features	IR
glass1	214	9	1.82
haberman	306	3	2.78
new-thyroid1	215	5	5.14
pima	768	8	1.87
vehicle1	846	18	2.9
wisconsin	683	9	1.86
yeast3	1484	8	8.1
abalone9-18	731	8	16.4
abalone19	4174	8	129.44
ecoli4	336	7	15.8
shuttle-c2-vs-c4	129	9	20.9
vowel0	988	13	9.98
yeast4	1484	8	28.1
yeast5	1484	8	32.73
yeast-1_vs_7	459	7	14.3
ecoli-0-1_vs_2-3-5	244	7	9.17
ecoli-0-1_vs_5	240	6	11
poker-8-9_vs_5	2075	10	82
shuttle-6_vs_2-3	230	9	22
shuttle-2_vs_5	3316	9	66.67
winequality-red-4	1599	11	29.17
cleveland-0_vs_4	177	13	12.62

IV. EXPERIMENTS AND ANALYSIS

The experimental environment is shown as follows: the CPU used is Intel Xeon E5-1620 v3, 4 cores, 3.50GHz, and the memory is 8GB. The operating system is the windows 10, and the software platform is MATLAB R2020a.

In order to verify the effectiveness of the OWA-ELM algorithm, the algorithm was compared with the ELM, WELM1, WELM2, CCR-ELM and CS-ELM algorithms on 22 imbalance binary classification data sets (WELM1 and WELM2 are WELM algorithms using weighting scheme W1 and W2 respectively).

A. PARAMETER SETTINGS

To ensure the reliability and accuracy of the experiments, each algorithm used ten times of five-fold cross-validation on each data set, and takes the value of average and standard deviation as the final result. For each algorithm, the grid cross search method was used to obtain the best parameter combination. In this step, five-fold cross validation was also used to ensure that the obtained parameter combination is the best. the search range of the number of hidden layer neurons L of all algorithms was [10, 20, …, 190, 200]. The search range of the regularization parameter C of ELM, WELM1, WELM2, CS-ELM and OWA-ELM was [2^{-20}, 2^{-18}, …, 2^{18}, 2^{20}]. The search range of the two regularization parameters C^+ and C^- in CCR-ELM was also [2^{-20}, 2^{-18}, …, 2^{18}, 2^{20}]. The search range of OWA-ELM increment value Δ was [0.001, 0.004, 0.007, …, 0.03].

B. DATASETS

The data sets used in the experiments are 22 binary data sets with different range of imbalance rates downloaded from the KEEL data set repository. The detailed information of the data sets is shown in Table 1. The index to measure the degree of data imbalance is the imbalance rate, and its calculation

TABLE 2. Comparison of G-mean ± std values of each method on 22 datasets (sigmoid node).

Dataset	ELM	WELM1	WELM2	CCR-ELM	CS-ELM	OWA-ELM
	mean ± std (C, L)	mean ± std (C, L)	mean ± std (C, L)	mean ± std (C^+, C^-, L)	mean ± std (C, L)	mean ± std (C, L, Δ)
glass1	0.6781±0.0862 (2^{20}, 200)	0.6124±0.1068 (2^{4}, 130)	0.6313±0.0757 (2^{18}, 200)	0.6677±0.0870 (2^{18}, 2^{16}, 180)	0.6756±0.0685 (2^{20}, 200)	**0.6912±0.0856** (2^{20}, 200, 0.001)
haberman	0.4929±0.0851 (2^{14}, 120)	0.5833±0.0651 (2^{12}, 190)	0.5925±0.0572 (2^{18}, 170)	0.4724±0.0879 (2^{10}, 2^{8}, 140)	0.6269±0.0846 (2^{-4}, 130)	**0.6362±0.0684** (2^{0}, 130, 0.013)
new-thyroid1	0.9459±0.0793 (2^{6}, 120)	0.9449±0.0673 (2^{18}, 170)	**0.9855±0.0183** (2^{10}, 200)	0.9034±0.1202 (2^{4}, 2^{10}, 130)	0.9281±0.0863 (2^{-6}, 130)	0.9477±0.0765 (2^{4}, 90, 0.016)
pima	0.5692±0.0456 (2^{18}, 180)	0.6237±0.0368 (2^{2}, 50)	0.5502±0.0456 (2^{20}, 40)	0.5785±0.0496 (2^{18}, 2^{6}, 190)	0.6317±0.0442 (2^{-2}, 160)	**0.6415±0.0368** (2^{-6}, 170, 0.007)
vehicle1	0.5236±0.0811 (2^{8}, 160)	0.6591±0.0354 (2^{14}, 70)	0.6747±0.0323 (2^{12}, 170)	0.5074±0.0813 (2^{14}, 2^{12}, 180)	**0.6874±0.0410** (2^{8}, 190)	0.6806±0.0455 (2^{2}, 190, 0.007)
wisconsin	0.9503±0.0160 (2^{-4}, 170)	0.9207±0.0288 (2^{18}, 190)	0.9405±0.0200 (2^{20}, 40)	0.9513±0.0180 (2^{0}, 2^{20}, 180)	0.9497±0.0191 (2^{4}, 120)	**0.9552±0.0226** (2^{0}, 150, 0.001)
yeast3	0.7906±0.0304 (2^{16}, 200)	0.9222±0.0234 (2^{20}, 140)	0.9247±0.0219 (2^{20}, 50)	0.7924±0.0474 (2^{20}, 2^{18}, 160)	**0.9263±0.0208** (2^{12}, 110)	0.9181±0.0292 (2^{14}, 100, 0.019)
abalone9-18	0.5308±0.1437 (2^{20}, 120)	0.8258±0.0723 (2^{18}, 120)	0.8206±0.0568 (2^{14}, 190)	0.5253±0.1555 (2^{18}, 2^{20}, 100)	0.8297±0.0732 (2^{10}, 190)	**0.8609±0.0653** (2^{8}, 120, 0.019)
abalone19	0.0000± 0.0000 (-, -)	0.7429±0.0959 (2^{10}, 50)	0.7262±0.0950 (2^{18}, 100)	0.0000±0.0000 (-, -, -)	**0.7612±0.0812** (2^{2}, 110)	0.5574±0.2592 (2^{-2}, 200, 0.016)
ecoli4	0.8889±0.1057 (2^{16}, 60)	0.9445±0.0594 (2^{20}, 20)	0.9508±0.0472 (2^{14}, 10)	0.8324±0.1783 (2^{20}, 2^{12}, 40)	0.9298±0.0796 (2^{12}, 30)	**0.9580±0.0654** (2^{8}, 130, 0.013)
shuttle-c2-vs-c4	0.8411±0.3066 (2^{-4}, 160)	0.9321±0.2415 (2^{12}, 60)	**0.9376±0.2394** (2^{2}, 80)	0.6624±0.4642 (2^{-4}, 2^{-2}, 150)	0.8838±0.2143 (2^{-8}, 160)	0.8347±0.3498 (2^{-4}, 70, 0.013)
vowel0	0.9983±0.0081 (2^{4}, 200)	**0.9993±0.0016** (2^{20}, 160)	0.9890±0.0068 (2^{20}, 90)	0.9926±0.0158 (2^{2}, 2^{8}, 190)	0.9980±0.0056 (2^{12}, 200)	0.9961±0.0123 (2^{6}, 190, 0.001)
yeast4	0.3895±0.0616 (2^{18}, 190)	0.8288±0.0716 (2^{6}, 50)	0.8089±0.0538 (2^{20}, 10)	0.3162±0.1839 (2^{16}, 2^{20}, 150)	**0.8371±0.0533** (2^{0}, 120)	0.8216±0.0671 (2^{-4}, 100, 0.028)
yeast5	0.5971±0.1304 (2^{20}, 180)	0.9563±0.0252 (2^{20}, 80)	0.9548±0.0218 (2^{20}, 190)	0.5872±0.1730 (2^{20}, 2^{18}, 190)	0.9599±0.0214 (2^{16}, 80)	**0.9612±0.0353** (2^{6}, 130, 0.016)
yeast-1_vs_7	0.4144±0.2451 (2^{20}, 100)	0.7338±0.0827 (2^{18}, 80)	0.7378±0.0740 (2^{18}, 40)	0.4255±0.2408 (2^{20}, 2^{18}, 110)	0.7596±0.0864 (2^{8}, 100)	**0.7615±0.0867** (2^{2}, 90, 0.028)
ecoli-0-1_vs_2-3-5	0.7848±0.1442 (2^{0}, 100)	0.8193±0.1233 (2^{2}, 30)	0.7078±0.1713 (2^{4}, 10)	0.7696±0.1546 (2^{12}, 2^{4}, 170)	**0.8799±0.0993** (2^{0}, 140)	0.8768±0.0812 (2^{-2}, 170, 0.013)
ecoli-0-1_vs_5	0.7626±0.1245 (2^{10}, 180)	0.8411±0.1288 (2^{18}, 120)	0.8239±0.1553 (2^{12}, 170)	0.7946±0.1949 (2^{6}, 2^{8}, 190)	0.8545±0.1027 (2^{4}, 150)	**0.9014±0.0921** (2^{-2}, 120, 0.013)
poker-8-9_vs_5	0.0089±0.0632 (2^{18}, 40)	0.6212±0.1527 (2^{20}, 40)	**0.6802±0.1538** (2^{14}, 170)	0.0089±0.0632 (2^{16}, 2^{14}, 160)	0.6591±0.1961 (2^{4}, 200)	0.6145±0.1696 (2^{2}, 140, 0.025)
shuttle-6_vs_2-3	**1.0000±0.0000** (2^{-8}, 100)	0.9920±0.0211 (2^{-14}, 90)	0.9998±0.0016 (2^{-2}, 170)	0.9683±0.1513 (2^{-8}, 2^{-8}, 150)	0.9970±0.0060 (2^{-20}, 190)	0.9937±0.0414 (2^{-14}, 100, 0.007)
shuttle-2_vs_5	0.9780±0.0371 (2^{4}, 160)	0.9872±0.0245 (2^{6}, 70)	0.9878±0.0247 (2^{14}, 160)	0.9635±0.0439 (2^{-4}, 2^{2}, 170)	**0.9944±0.0192** (2^{0}, 170)	0.9923±0.0183 (2^{-4}, 160, 0.01)
winequality-red-4	0.0544±0.1173 (2^{20}, 140)	**0.5574±0.0900** (2^{8}, 160)	0.5530±0.0923 (2^{12}, 180)	0.0244±0.0835 (2^{16}, 2^{14}, 200)	0.5258±0.1095 (2^{8}, 200)	0.5049±0.1421 (2^{0}, 180, 0.019)
cleveland-0_vs_4	0.2502±0.3117 (2^{20}, 170)	**0.5534±0.2705** (2^{14}, 40)	0.5319±0.2919 (2^{14}, 170)	0.1882±0.2940 (2^{10}, 2^{14}, 170)	0.5527±0.3194 (2^{0}, 100)	0.5403±0.2815 (2^{-2}, 120, 0.025)

formula is as follows:

$$IR = \frac{\#(N_{maj})}{\#(N_{min})} \tag{16}$$

Here, $\#(N_{maj})$ is the number of majority instances, $\#(N_{min})$ is the number of minority instances. The higher the IR value, the more imbalance the data.

C. EVALUATION METHOD

Besides the comparison in training time, the main evaluation methods used in the experiment are G-mean and F-measure. Both G-mean and F-measure can provide a good comprehensive evaluation of the models used to classify imbalanced data. Their calculation formula is as follows:

$$G - mean = (\prod_{i=1}^{m} Acci)^{1/m} \tag{17}$$

$$F - measure = \frac{(\alpha + 1)Precision \times Recall}{\alpha^2(Precision + Recall)} \tag{18}$$

where, Acc_i represents the classification accuracy of samples with class i, $i = 1, 2, \ldots,$ m. *Precision* and *Recall* are shown in (19) and (20) respectively.

$$Precision = \frac{TP}{TP + FP} \tag{19}$$

$$Recall = \frac{TP}{TP + FN} \tag{20}$$

TABLE 3. Comparison of F-measure ± std values OF each method on 22 datasets (sigmoid node).

Dataset	ELM	WELM1	WELM2	CCR-ELM	CS-ELM	OWA-ELM
	mean ± std (C, L)	mean ± std (C, L)	mean ± std (C, L)	mean ± std (C^+, C^-, L)	mean ± std (C, L)	mean ± std (C, L, Δ)
glass1	0.5973±0.1051 (2^{18}, 120)	0.6038±0.0808 (2^{12}, 130)	0.6041±0.0569 (2^{20}, 170)	0.5900±0.1118 (2^{16}, 2^{20}, 200)	0.6006±0.0907 (2^{20}, 110)	**0.6179±0.0796** (2^{20}, 180, 0.004)
haberman	0.3382±0.1434 (2^{8}, 120)	0.4721±0.0847 (2^{8}, 80)	0.4208±0.0775 (2^{14}, 180)	0.3164±0.1264 (2^{8}, 2^{14}, 80)	0.4895±0.0833 (2^{-6}, 110)	**0.5113±0.0824** (2^{-4}, 120, 0.016)
new-thyroid1	0.8945±0.0950 (2^{12}, 130)	0.9079±0.1058 (2^{14}, 120)	0.8586±0.1406 (2^{14}, 110)	0.9165±0.0893 (2^{4}, 2^{2}, 200)	0.9196±0.0864 (2^{6}, 140)	**0.9212±0.0929** (2^{6}, 200, 0.001)
pima	0.4592±0.0666 (2^{18}, 190)	0.5364±0.0427 (2^{10}, 140)	0.5610±0.0395 (2^{20}, 90)	0.4322±0.0763 (2^{20}, 2^{0}, 140)	0.5580±0.0434 (2^{-2}, 160)	**0.5712±0.0387** (2^{-6}, 140, 0.01)
vehicle1	0.3962±0.0754 (2^{16}, 200)	0.5326±0.0487 (2^{6}, 130)	0.5078±0.0374 (2^{18}, 100)	0.3508±0.0899 (2^{12}, 2^{12}, 160)	**0.5385±0.0593** (2^{0}, 190)	0.5233±0.0492 (2^{4}, 110, 0.016)
wisconsin	0.9079±0.0278 (2^{4}, 70)	0.9219±0.0317 (2^{14}, 40)	0.9273±0.0207 (2^{12}, 70)	0.9389±0.0277 (2^{18}, 2^{-2}, 150)	0.9380±0.0250 (2^{-2}, 180)	0.9331±0.0195 (2^{0}, 140, 0.004)
yeast3	0.7066±0.0635 (2^{20}, 120)	0.7039±0.0508 (2^{20}, 20)	0.6740±0.0320 (2^{20}, 90)	0.7158±0.0556 (2^{18}, 2^{20}, 130)	0.7310±0.0464 (2^{18}, 40)	**0.7810±0.0491** (2^{10}, 140, 0.007)
abalone9-18	0.3758±0.1203 (2^{14}, 130)	0.5036±0.0831 (2^{16}, 30)	0.4038±0.0683 (2^{14}, 190)	0.3932±0.1658 (2^{14}, 2^{18}, 180)	0.5136±0.0945 (2^{8}, 90)	**0.6587±0.1152** (2^{4}, 120, 0.013)
abalone19	0.0000± 0.0000 (-, -)	**0.0480±0.0126** (2^{16}, 30)	0.0427±0.0085 (2^{20}, 110)	0.0000±0.0000 (-, -, -)	0.0469±0.0170 (2^{20}, 130)	0.0325±0.0523 (2^{6}, 160, 0.019)
ecoli4	0.8600±0.1616 (2^{16}, 150)	0.7486±0.1177 (2^{18}, 30)	0.7122±0.1441 (2^{16}, 190)	0.8411±0.1473 (2^{10}, 2^{10}, 100)	0.7655±0.1154 (2^{14}, 80)	**0.8713±0.1309** (2^{16}, 30, 0.022)
shuttle-c2-vs-c4	0.8733±0.2848 (2^{-4}, 180)	0.9333±0.2428 (2^{2}, 200)	**0.9560±0.1116** (2^{6}, 160)	0.8600±0.3305 (2^{-4}, 2^{0}, 200)	0.7674±0.3384 (2^{-6}, 130)	0.8767±0.2873 (2^{-6}, 140, 0.013)
vowel0	0.9890±0.0158 (2^{2}, 160)	0.9904±0.0168 (2^{18}, 170)	0.9893±0.0170 (2^{16}, 180)	0.9854±0.0207 (2^{2}, 2^{8}, 200)	0.9910±0.0173 (2^{8}, 190)	**0.9967±0.0090** (2^{6}, 190, 0.001)
yeast4	0.2405±0.1143 (2^{18}, 140)	0.2675±0.0484 (2^{16}, 180)	0.2182±0.0240 (2^{10}, 90)	0.2206±0.1387 (2^{20}, 2^{20}, 100)	0.2787±0.0453 (2^{18}, 200)	**0.4194±0.1350** (2^{16}, 70, 0.022)
yeast5	0.4357±0.1684 (2^{16}, 190)	0.4903±0.0582 (2^{18}, 80)	0.4493±0.0423 (2^{20}, 140)	0.4805±0.1531 (2^{18}, 2^{20}, 150)	0.4869±0.0606 (2^{20}, 170)	**0.6401±0.0898** (2^{18}, 120, 0.01)
yeast-1_vs_7	0.3664±0.1609 (2^{18}, 170)	0.2869±0.0747 (2^{20}, 200)	0.2531±0.0440 (2^{14}, 170)	0.3580±0.2059 (2^{20}, 2^{20}, 170)	0.3158±0.0649 (2^{12}, 60)	**0.4295±0.1886** (2^{20}, 110, 0.007)
ecoli-0-1_vs_2-3-5	**0.7142±0.1276** (2^{-2}, 110)	0.5416±0.1525 (2^{12}, 130)	0.5920±0.1817 (2^{10}, 170)	0.6833±0.1799 (2^{10}, 2^{0}, 160)	0.6550±0.1541 (2^{2}, 190)	0.7342±0.1793 (2^{-4}, 150, 0.007)
ecoli-0-1_vs_5	0.7778±0.1443 (2^{-2}, 90)	0.6745±0.1787 (2^{6}, 80)	0.5901±0.1284 (2^{18}, 140)	0.7316±0.1627 (2^{0}, 2^{18}, 150)	0.7607±0.1513 (2^{2}, 190)	**0.7780±0.1603** (2^{-4}, 200, 0.004)
poker-8-9_vs_5	0.0067±0.0471 (2^{8}, 30)	0.0515±0.0193 (2^{18}, 30)	0.0938±0.0465 (2^{18}, 190)	0.0333±0.1010 (2^{18}, 2^{18}, 130)	**0.1128±0.0483** (2^{14}, 200)	0.0925±0.1386 (2^{16}, 200, 0.013)
shuttle-6_vs_2-3	**1.0000±0.0000** (2^{-8}, 90)	0.9507±0.0969 (2^{-20}, 200)	0.9840±0.0548 (2^{-4}, 120)	0.9800±0.0800 (2^{-8}, 2^{-8}, 170)	0.9501±0.1070 (2^{-20}, 170)	0.9667±0.1543 (2^{-14}, 100, 0.07)
shuttle-2_vs_5	0.9711±0.0451 (2^{2}, 170)	0.8936±0.0814 (2^{20}, 140)	0.9650±0.0482 (2^{8}, 140)	0.9654±0.0462 (2^{2}, 2^{6}, 150)	0.9706±0.0462 (2^{0}, 150)	**0.9772±0.0358** (2^{-2}, 200, 0.01)
winequality-red-4	0.0325±0.0621 (2^{14}, 170)	0.0913±0.0329 (2^{-14}, 120)	0.0867±0.0243 (2^{12}, 180)	0.0237±0.0625 (2^{20}, 2^{16}, 180)	**0.1007±0.0297** (2^{2}, 190)	0.0581±0.0716 (2^{0}, 140, 0.019)
cleveland-0_vs_4	0.1415±0.1777 (2^{20}, 140)	0.1808±0.1561 (2^{18}, 100)	0.1836±0.0954 (2^{16}, 50)	0.1457±0.2301 (2^{12}, 2^{12}, 190)	**0.2444±0.1517** (2^{0}, 140)	0.1484±0.1849 (2^{8}, 200, 0.004)

TP and *TN* respectively represent the number of samples of the minority class and the majority class that were correctly classified, while *FP* and *FN* represent the number of samples of the majority class and the minority class that were misclassified, respectively.

In the experiment, the normal form of F-measure was set to 1, that is, α in (18) is equal to 1. Since the data set used in the experiment is a binary data set, G-mean can also be expressed as:

$$G - mean = \sqrt{Recall \times Specificity} \tag{21}$$

The calculation formula of Specificity is as follows:

$$Specificity = \frac{TN}{TN + FP} \tag{22}$$

D. EXPERIMENTAL RESULTS

Table 2 and Table 3 show the experimental results of each algorithm on 22 data sets. The results marked in bold indicate the optimal results, while the results marked with underline denote the suboptimal. From the experimental results we can know:

1. ELM can't classify imbalanced data well, especially some highly imbalanced data. In the comparison of G-mean, compared to WELM1, WEWLM2, CCR-ELM, CS-ELM and OWA-ELM, the results obtained by ELM on most data sets are the worst, and the comparison of F-measure evaluation is not as good as WELM1, WEWLM2, CS-ELM and OWA-ELM. On data sets such as abalone19, poker-8-9_vs_5, and winequality-red-4 with large imbalance rates, the G-mean and F-measure values of ELM are even equal to or close to zero. This is mainly because, like traditional classification algorithms, Extreme Learning Machine was not designed to handle imbalanced data, which makes it easy to ignore minority samples. However, on the shuttle-2_vs_5 dataset with a high imbalance rate, like other algorithms, ELM also achieved good results. Perhaps the data distribution of this data set is relatively simple, and all algorithms can find the appropriate decision boundary very well.

2. The performance of CCR-ELM on G-mean and F-measure is inferior to WELM1, WELM2, CS-ELM and OWA-ELM, and it is only slightly better than ELM on G-mean. This is because, in the solution of $\boldsymbol{\beta}$ of CCR-ELM (12), if C^+ and C^- are not in the same order of magnitude, for example, C^+ is equal to 2^{10} and C^- is equal to 2^2, then $\frac{\boldsymbol{I}}{C^+} + \frac{\boldsymbol{I}}{C^-} \approx \frac{\boldsymbol{I}}{C}$. As a result, the solution formula of $\boldsymbol{\beta}$ of CCR-ELM is almost the same as that of ELM.

3. The performance of OWA-ELM algorithm in experiments was better than other algorithms, especially in F-measure evaluation. In the G-mean evaluation, 9 times optimal and 4 times suboptimal results were obtained. In F-measure, 13 times optimal and one suboptimal results were obtained. This is due to the characteristics of output weight matrix adjustment in OWA-ELM. By adding a reasonable increment Δ to the connection weights between hidden layer neurons and minority output neurons in ELM, the output value of the minority output neuron is increased, so the probability of the minority samples being correctly predicted is increased. Especially for the minority samples in the overlapping area, the probability that is correctly classified increases. The excellent performance of the OWA-ELM algorithm also confirms that a slight increment in the value of all elements in a certain column of the output weight matrix will not cause obvious structural risks to the ELM neural network or resulting in overfitting, but will enable the algorithm better to classify imbalanced data.

4. In addition to the imbalance rate, there are other factors that affect the classifier to classify imbalanced data. On some data sets with low imbalance rate, such as glass1, haberman, pima, and vehicle1, the G-mean value of each algorithm is between 0.5 and 0.7, and the F-measure evaluation value of each algorithm does not exceed 0.62. But on some highly imbalanced data sets, such as the shuttle-2_vs_5 data set that with an imbalance rate of 66.67, better results have been achieved. This is because, besides the imbalance rate, the class imbalance problem also includes other factors, such as class overlapping and disjuncts [24]. Perhaps the data sets such as glass1, haberman, pima, and vehicle1 have serious class overlapping or disjuncts, which affect the classification performance of each algorithm.

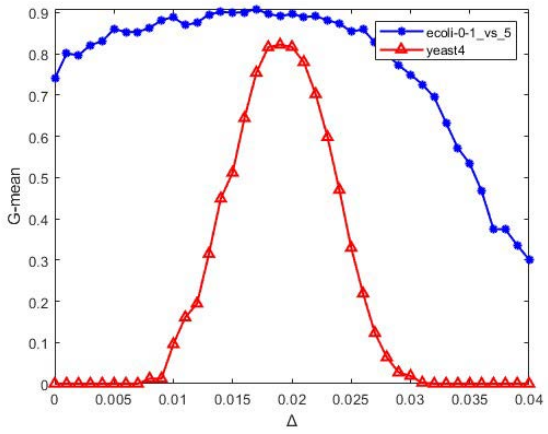

FIGURE 2. The effect of Δ on G-mean. ($C_{ecoli} = 2^{-2}$, $L_{ecoli} = 150$, $C_{yeast4} = 2^{18}$, $L_{yeast4} = 180$).

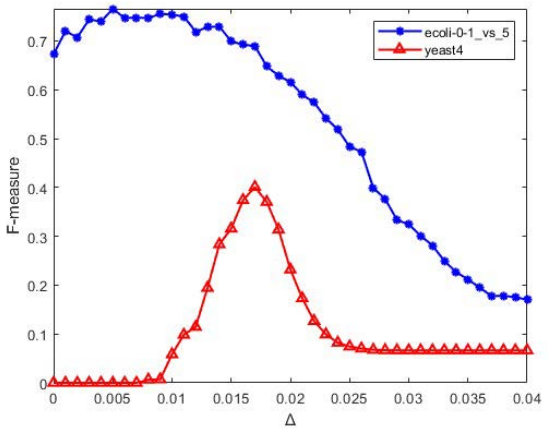

FIGURE 3. The effect of Δ on F-measure. ($C_{ecoli} = 2^{-4}$, $L_{ecoli} = 180$, $C_{yeast4} = 2^{20}$, $L_{yeast4} = 170$).

E. THE IMPACT OF INCREMENT VALUE (Δ)

Besides the regularization parameter C and the number of hidden layer nodes L, another important parameter of OWA-ELM algorithm is the increment value (Δ). For this reason, an experiment on the effect of increment value (Δ) on G-mean, F-measure, sensitivity and specificity was carried out on ecoli-0-1_vs_5 and yeast4 data sets. The experimental results are shown in figure 2-4.

It can be seen from figure 2 and figure 3 that as Δ starts to increase from 0, the G-mean and F-measure values of ecoli-0-1_vs_5 and yeast4 data sets are increasing, and begin to decrease after increasing to the maximum value. Even, in the test on the yeast4 data set, G-mean even increased from 0 to more than 0.7, which means that OWA-ELM has achieved better results than ELM on this data set in G-mean. This is due to the output weight adjustment strategy of OWA-ELM. As shown in figure 4, when Δ gradually increases to a suitable value, the sensitivity value increases, while the specificity value decreases slightly. This means

TABLE 4. Comparison of training time of each method on 22 datasets (sigmoid node) (unit: ms).

Dataset	ELM	WELM1	WELM2	CCR-ELM	CS-ELM	OWA-ELM
glass1	1.7769	2.2627	2.0404	1.8742	2.4997	1.7879
haberman	2.0595	2.4336	2.3359	2.1502	2.6262	2.0727
new-thyroid1	1.7313	2.1377	2.0871	1.8534	2.4649	1.7164
pima	3.2283	5.5246	5.4163	3.3774	4.0796	3.2462
vehicle1	6.1549	8.8940	8.9284	6.2664	6.6055	6.2300
wisconsin	3.1178	4.8727	4.8051	3.3049	3.8866	3.1573
yeast3	5.6054	14.2675	13.5135	6.0700	7.8832	5.7748
abalone9-18	3.3823	5.6833	5.2429	3.7680	4.1391	3.4739
abalone19	13.2602	83.0256	84.7005	14.4045	18.6643	13.5750
ecoli4	2.0732	2.3618	2.2324	2.1788	2.3820	2.0608
shuttle-c2-vs-c4	1.3414	1.5502	1.5537	1.4318	1.7913	1.4642
vowel0	4.6558	8.7410	8.8693	5.0865	6.3181	4.6994
yeast4	5.6368	14.0959	13.7274	6.0483	7.8456	5.7153
yeast5	5.6164	15.0493	14.6654	6.3186	7.8795	5.6215
yeast-1_vs_7	2.5326	3.2635	3.0621	2.6986	3.1233	2.5323
ecoli-0-1_vs_2-3-5	1.7593	2.1087	2.2120	1.9521	2.4061	1.8233
ecoli-0-1_vs_5	1.7929	2.0367	2.0463	1.8023	2.3961	1.8301
poker-8-9_vs_5	7.4361	24.4707	24.2120	8.2166	11.3479	7.4558
shuttle-6_vs_2-3	1.8567	2.1311	2.0362	1.8555	2.3215	1.9309
shuttle-2_vs_5	10.8287	57.6100	56.4443	11.4658	15.9284	11.3758
winequality-red-4	6.0717	16.3663	16.4855	6.6021	9.2560	6.4598
cleveland-0_vs_4	1.6502	1.7490	1.8704	1.8354	1.9325	1.7739

that the classification accuracy of the minority samples is improved without significantly affecting the accuracy of the classification of the majority samples. So, the G-mean value is improved.

As for the selection of Δ, if it is too small or even close to 0, OWA-ELM will degenerate into ELM; if it is too large, it will lead to a significant decline in the classification accuracy of majority samples. In the experiment, a more appropriate value is obtained through a 5-fold cross search, and the search range is [0.001:0.003:0.03], which is also a more appropriate and recommended Δ range. In the test of the G-mean value and F-measure value on 22 data sets, it is also found that the optimal value of Δ on more than half of the data sets is in the range of 0.01 to 0.02.

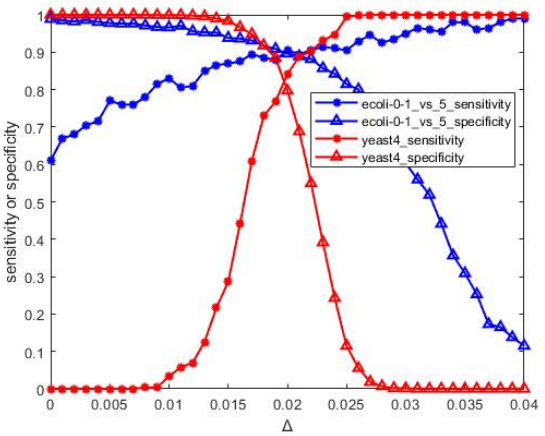

FIGURE 4. The effect of Δ on sensitivity and specificity. ($C_{ecoli} = 2^{-2}$, $L_{ecoli} = 150$, $C_{yeast4} = 2^{18}$, $L_{yeast4} = 180$).

F. TRAINING TIME COMPARISON

In terms of time comparison, we mainly compared the training time of ELM, WELM1, WEWLM2, CCR-ELM, CS-ELM and OWA-ELM. In order to make the results more reliable, the control variates method was adopted. The number of hidden layer neurons in each algorithm was 200. In ELM, WELM1, WELM2, CS-ELM and OWA-ELM, the regularization parameter C was 2^4. In CCR-ELM, C^+ and C^- were both 2^4. In OWA-ELM, the Δ was 0.003. Table 4 shows the comparison results. It can be seen from Table 4 that ELM has the least average training time on 22 training sets, followed by OWA-ELM and CCR-ELM. The training time of OWA-ELM is almost equivalent to that of ELM. However, WELM requires the most training time.

This is because the OWA-ELM adjusts $\boldsymbol{\beta}$ during training, which increases the training time compared to ELM. Although CCR-ELM, CS-ELM and OWA-ELM perform addition operations on the matrix, but the difference is that CCR-ELM needs to calculate ($\boldsymbol{I}/C^+ + \boldsymbol{I}/C^-$) in (13) and the CS-ELM requires more matrix operations to solve $\boldsymbol{\beta}$. Although OWA-ELM adds $\boldsymbol{\beta}$ after obtaining the output weight, calculation in this step is less complex, and it takes less time. These two aspects also cause the training time of OWA-ELM to be almost equal to the training time of ELM, and even the training time required on most data sets is slightly shorter than that of CCR-ELM. CS-ELM also needs more time to train, because the operation of its output weight matrix is also more complicated. Comparing (11) and (5), it can be found that compare with ELM, two matrix multiplication operations are added during the solution process of $\boldsymbol{\beta}$ of WELM. The matrix multiplication operation is more time-consuming than the matrix addition operation, which directly leads to the training time of WELM more than that of other algorithms.

However, since OWA-ELM adds a parameter Δ, it takes more time than ELM and WELM to search for the optimal parameter combination in grid cross search, and it takes more time to search for the optimal parameter combination. One

possible way is to fully analyze the possible range of Δ and avoid searching for Δ in meaningless ranges. Another feasible method is to use the powerful ability of swarm intelligence algorithms in finding hyperparameters to search for the best combination of parameters in Δ, C and L, which may not take much time.

V. CONCLUSION

ELM has attracted the attention and use of various industries because of its short training time and good generalization performance. However, in practical applications, the data is often imbalanced. When dealing with imbalanced data sets, the performance of ELM is often not good. Therefore, it is proposed an improved ELM algorithm with output weight adjustment (OWA-ELM). By slightly increasing the corresponding column in the output weight, that is, slightly increasing the weight between hidden layer neurons and the minority output neuron to increase the output of minority output neuron. After that, and the accuracy of minority class samples increases without significantly affecting the classification of the majority samples. Experiments on 22 KEEL imbalanced data sets show that in terms of G-mean and F-measure evaluation, the proposed OWA-ELM algorithm has excellent performance on most data sets, especially in F-measure evaluation. In the future, we will focus on optimizing the performance of OWA-ELM in the classification of imbalanced data sets with overlapping and disjuncts, and shorten the time required to find the optimal combination of parameters. In addition, applying the proposed algorithm to the online imbalanced data classification problem is also an important work in future.

REFERENCES

[1] S. Dhar and V. Cherkassky, "Development and evaluation of cost-sensitive Universum-SVM," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 806–818, Apr. 2015.
[2] A. Teng, L. Peng, Y. Xie, H. Zhang, and Z. Chen, "Gradient descent evolved imbalanced data gravitation classification with an application on internet video traffic identification," *Inf. Sci.*, vol. 539, pp. 447–460, Oct. 2020.
[3] Z. Li, K. Kamnitsas, and B. Glocker, "Analyzing overfitting under class imbalance in neural networks for image segmentation," *IEEE Trans. Med. Imag.*, vol. 40, no. 3, pp. 1065–1077, Mar. 2021.
[4] C. Huang, X. Huang, Y. Fang, J. Xu, Y. Qu, P. Zhai, L. Fan, H. Yin, Y. Xu, and J. Li, "Sample imbalance disease classification model based on association rule feature selection," *Pattern Recognit. Lett.*, vol. 133, pp. 280–286, May 2020.
[5] C. V. Priscilla and D. P. Prabha, "Influence of optimizing XGBoost to handle class imbalance in credit card fraud detection," in *Proc. 3rd Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Aug. 2020, pp. 1309–1315.
[6] Q. Liu, G. Ma, and C. Cheng, "Data fusion generative adversarial network for multi-class imbalanced fault diagnosis of rotating machinery," *IEEE Access*, vol. 8, pp. 70111–70124, 2020.
[7] N. Tomašev and D. Mladenić, "Class imbalance and the curse of minority hubs," *Knowl.-Based Syst.*, vol. 53, pp. 157–172, Nov. 2013.
[8] Z. Xu, D. Shen, T. Nie, and Y. Kou, "A hybrid sampling algorithm combining M-SMOTE and ENN based on random forest for medical imbalanced data," *J. Biomed. Informat.*, vol. 107, Jul. 2020, Art. no. 103465.
[9] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.
[10] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern., B, Cybern.*, vol. 42, no. 2, pp. 513–524, Apr. 2012.
[11] B. Ahuja and V. P. Vishwakarma, "Local feature extraction based KELM for face recognition," in *Proc. 12th Int. Conf. Contemp. Comput. (IC3)*, Aug. 2019, pp. 1–5.
[12] K. Rujirakul and C. So-In, "Histogram equalized deep PCA with ELM classification for expressive face recognition," in *Proc. Int. Workshop Adv. Image Technol. (IWAIT)*, Jan. 2018, pp. 1–4.
[13] X. Ximeng, Y. Rennong, and Y. Yang, "Threat assessment in air combat based on ELM neural network," in *Proc. IEEE Int. Conf. Artif. Intell. Comput. Appl. (ICAICA)*, Mar. 2019, pp. 114–120.
[14] F. Huang, J. Lu, J. Tao, L. Li, X. Tan, and P. Liu, "Research on optimization methods of ELM classification algorithm for hyperspectral remote sensing images," *IEEE Access*, vol. 7, pp. 108070–108089, 2019.
[15] L. Li, J. Zeng, L. Jiao, P. Liang, F. Liu, and S. Yang, "Online active extreme learning machine with discrepancy sampling for PolSAR classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 3, pp. 2027–2041, Mar. 2020.
[16] M. Sharifmoghadam and H. Jazayeriy, "Breast cancer classification using AdaBoost-extreme learning machine," in *Proc. 5th Iranian Conf. Signal Process. Intell. Syst. (ICSPIS)*, Dec. 2019, pp. 1–5.
[17] S. Saxena, S. Shukla, and M. Gyanchandani, "Breast cancer histopathology image classification using kernelized weighted extreme learning machine," *Int. J. Imag. Syst. Technol.*, vol. 31, no. 1, pp. 168–179, Mar. 2021.
[18] K. Li, X. Kong, Z. Lu, L. Wenyin, and J. Yin, "Boosting weighted elm for imbalanced learning," *Neurocomputing*, vol. 128, pp. 15–21, Mar. 2014.
[19] W. Zong, G.-B. Huang, and L. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, Feb. 2013.
[20] H. Yu, C. Sun, X. Yang, S. Zheng, Q. Wang, and X. Xi, "LW-ELM: A fast and flexible cost-sensitive learning framework for classifying imbalanced data," *IEEE Access*, vol. 6, pp. 28488–28500, 2018.
[21] W. Xiao, J. Zhang, Y. Li, S. Zhang, and W. Yang, "Class-specific cost regulation extreme learning machine for imbalanced classification," *Neurocomputing*, vol. 261, pp. 70–82, Oct. 2017.
[22] B. S. Raghuwanshi and S. Shukla, "Class-specific extreme learning machine for handling binary class imbalance problem," *Neural Netw.*, vol. 105, pp. 206–217, Sep. 2018.
[23] W. Mao, M. Jiang, J. Wang, and Y. Li, "Online extreme learning machine with hybrid sampling strategy for sequential imbalanced data," *Cognit. Comput.*, vol. 9, no. 6, pp. 780–800, Aug. 2017.
[24] H. Kaur, H. S. Pannu, and A. K. Malhi, "A systematic review on imbalanced data challenges in machine learning: Applications and solutions," *ACM Comput. Surv.*, vol. 52, no. 4, pp. 1–36, Jul. 2020.

XIAOPENG ZHANG was born in Guangdong, China, in 1995. He is currently studying for a master's degree at Guangxi University. His research interests include machine learning, data mining, and swarm intelligence algorithm.

LIANGXI QIN was born in 1963. He received his Ph.D. degree in computer software and theory from the Graduate School of Chinese Academy of Sciences, in June 2005. He is currently a Professor in the School of Computer, Electronics and Information, Guangxi University. His research interests include data mining, machine learning, and decision rough set.