Class-specific extreme learning machine based on overall distribution for addressing binary imbalance problem

Bhagat Singh Raghuwanshi (✉ bhagat.mnit@gmail.com)

Madhav Institute of Technology and Science <https://orcid.org/0000-0002-3027-7831>

Research Article

Keywords: Class imbalance learning, Extreme learning machine, Classification

Posted Date: February 21st, 2022

DOI: <https://doi.org/10.21203/rs.3.rs-1168985/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Class-specific extreme learning machine based on overall distribution for addressing binary imbalance problem

Bhagat Singh Raghuwanshi*

Received: date / Accepted: date

Abstract Imbalanced classification problems have significantly unequal class proportions. Conventional extreme learning machine (ELM) gives the same importance to all the samples leading to the results which favor the majority class. To solve this intrinsic deficiency, modification of ELM have been developed like weighted ELM (WELM), WELM based on the overall distribution (ODW-ELM) etc.. We have also recently developed class-specific ELM (CS-ELM) in one of our recent works. It has been shown in this work that the derivation of the output weights, β is more efficient compared to class-specific cost regulation ELM (CCRELM) for handling the class imbalance problem. Motivated by CCR-ELM X. Luo et al. has proposed the classifier ODW-ELM which is also not efficient for the imbalance learning. In this work, a novel class-specific WELM based on overall distribution (OD-CSELM) and the kernelized version of OD-CSELM (OD-CSKELM) is proposed to address the binary class imbalance problem more effectively. OD-CSELM and OD-CSKELM are motivated by CS-ELM. In addition, the computational complexities of OD-CSELM and OD-CSKELM are significantly lower than WELM and kernelized WELM respectively. The proposed work is evaluated by using the benchmark real-world imbalanced datasets. The experimental results demonstrate that the proposed work gives good generalization performance in contrast with rest of the classifiers for class imbalance learning.

Keywords Class imbalance learning · Extreme learning machine · Classification

* Corresponding author.

Bhagat Singh Raghuwanshi

Assistant Professor in Department of Information Technology, Madhav Institute of Technology and Science (MITS) Gwalior, Madhya Pradesh, 474005, India

E-mail: bhagat@mitsgwalior.in

1 Introduction

Learning from imbalanced class distributions have drawn increasing consideration from data mining community and many classification algorithms have been presented [1, 2, 3]. Almost all the classification problems in real-world do not have uniform class distribution. The classes whose number of samples are below the average number of samples per class are termed as the minority classes. The classes whose number of samples are above the average number of samples per class are termed as the majority classes. Some examples of the imbalanced learning are fraud detection [4, 5], Software Defect Prediction [6], Cancer malignancy grading [7] etc. The problem associated with the class imbalance learning is that conventional classifiers usually misclassify most of the minority class samples as the majority class samples. For problems like fraud detection, the minority class detection has greater importance in contrast with the majority class. Therefore, a higher positive recall value for the minority class is appropriate for any imbalanced classification problem. During the last decades, various methods have been proposed to handle the class imbalanced problems [1, 8]. The methods available for the imbalanced classification problems [8] can be broadly categorized as data level methods, algorithmic level methods and class-specific methods. Data level methods like oversampling and undersampling [1, 9] alters the data space to reduce the impact of the class imbalance. The undersampling method randomly selects a fraction of data from the majority class samples and balances the data distribution at the cost of information loss. For example, EasyEnsemble and BalanceCascade[9] algorithms employ undersampling for balancing the dataset. Oversampling method randomly duplicates the samples in the minority classes in order to enhance the cardinality of the minority class. This may lead to over-fitting. The informed oversampling approach, on the other hand, generates synthetic minority class samples to balance the class distribution like synthetic minority over-sampling techniques (SMOTE) [10].

Algorithmic level methods directly modify the classifier design to address the imbalanced learning. The cost-sensitive methods assign more penalty for misclassifying the minority class samples with respect to the majority class samples, that is, misclassification of minority class samples is much more costly. For example, WELM [11] and WSVM [12]. To find the optimal solution they minimize the weighted cumulative error with respect to each sample.

Extreme learning machine [13] has drawn increasing attention among the researchers around the world. It is a single-hidden layer feedforward neural networks (SLFNs) with random weights between the input and the hidden layer. It computes the weights between the hidden layer and the output layer analytically utilizing the Moore-Penrose (MP) pseudoinverse. This makes ELM much faster compared to the standard neural networks which require great effort in hyper-parameter tuning. Standard ELM [14, 15] does not take into consideration the class imbalance problem viably. Several variants of ELM-like WELM [11], Boosting WELM [16], Regularized Weighted Circular Complex valued ELM [17], CCR-ELM [18] and CS-ELM [19], class-specific cost-sensitive boosting WELM [20], UnderBagging reduced kernelized WELM [21] etc. have been designed to address the imbalanced learning effectively.

As mentioned above, random weights are employed to map the input data to the

feature space. These weights remain unaltered amid the training phase. Due to this some of the samples usually at the decision boundary are misclassified in certain realizations. It has been stated in [22, 23] that ELM with the Gaussian kernel also referred as kernelized ELM (KELM) outclass ELM with the Sigmoid node for the small and medium range datasets. ELM with the Gaussian kernel is relatively sluggish on the larger datasets in contrast with the ELM with Sigmoid node. It is also stated in [22, 23] that for most of the datasets (small and medium range datasets) ELM with Gaussian kernel runs faster compared to ELM with Sigmoid node.

WELM [11] minimizes the weighted cumulative error with respect to each sample. WELM uses two weighting schemes to assign class-wise weights to the samples. These weighting schemes assign more weight to increase the impact of the minority class while diminishing the relative impact of the majority class. It has been stated in [24] that, the Lagrangian multiplier, α relating to each minority class sample must be larger in weight in contrast with the α of the majority class samples. WELM employs two weighting schemes to assign a class-wise weight to the samples. The proposed algorithm employs class-specific regularization parameters instead of assigning weights to the training samples.

As mentioned above, α corresponding to the minority class samples should be greater than that of majority class samples. So, this work strengthens the regularization parameter for the minority class sample in contrast with the majority class samples in order to give more importance to the minority class samples.

The proposed algorithm assign weights to the class instead of assigning weight to the training samples. As mentioned above, α_i corresponding to the minority class should be greater than that of majority class. So, this work strengthens the weight for the minority-class compared to the majority class in order to give more importance to the minority class.

The work in ODW-ELM [25] is an extension of CCR-ELM and WELM. It has been shown in our recently published work [19] that the derivation of the output weights, β is more efficient than class-specific cost regulation ELM (CCR-ELM) for the imbalanced learning. Motivated by CCR-ELM, X. Luo et al. has proposed the classifier ODW-ELM which is also not efficient for the imbalance learning. In this work, a novel class-specific WELM based on overall distribution (OD-CSELM) and the kernelized version of OD-CSELM (OD-CSKELM) is proposed to address the binary class imbalance problem more effectively. The proposed classifiers, OD-CSELM and OD-CSKELM are motivated by CS-ELM.

The rest of the paper is organized as follows: The section 2 elaborates the related work in detail. The section 3 explains the proposed work. The section 4 presents the experimental setup and the result analysis. The last section concludes the paper along with the future work.

2 Related work

2.1 Extreme learning machine

Extreme learning machine was first proposed by Huang et al. [13, 26] for generalized single-hidden layer feedforward neural networks. ELM is much faster compared to the standard neural networks which require iterative hyper-parameter tuning. ELM randomly allocates the input weights and the hidden layer biases. The output weights of ELM are determined analytically utilizing the Moore-Penrose pseudoinverse. Given N training samples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$. Here, input vector $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in R^n$ and its desired output $\mathbf{t}_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T \in R^m$. Here, T stands for transpose. Here, the number of the input neurons n is set equal to the number of the features, L and m are the number of the hidden neurons and the output neurons respectively. The number of output neurons is equal to the number of classes. The input weight matrix is represented by $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_j, \dots, \mathbf{u}_L]^T \in R^{L \times n}$ and the bias of the hidden neurons, $\mathbf{b} = [b_1, b_2, \dots, b_j, \dots, b_L]^T \in R^L$, here, $\mathbf{u}_j = [u_{j1}, u_{j2}, \dots, u_{jn}]$ are the weights connecting the j^{th} hidden neuron with the input neurons, b_j is the bias of the j^{th} hidden neuron. These weights remain unaltered over the training phase. The output of the hidden layer for i^{th} sample is given as follows:

$$\phi(\mathbf{x}_i) = G(\mathbf{U}\mathbf{x}_i + \mathbf{b}) \quad (1)$$

Here, $G(\cdot)$ represents the hidden layer activation function. The output of the hidden layer for all the training samples is represented by Φ as shown in the following Eq:

$$\Phi = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_L(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_L(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \dots & \phi_L(x_N) \end{bmatrix}_{N \times L} \quad (2)$$

In the original ELM, the following optimization problem is formulated.

$$\begin{aligned} &\text{Minimize: } \frac{1}{2} \|\beta\|^2 + \lambda \frac{1}{2} \sum_{i=1}^N \|\xi_i\|^2 \\ &\text{Subject to: } \phi(\mathbf{x}_i)\beta = \mathbf{t}_i^T - \xi_i^T, \quad i = 1, \dots, N \end{aligned} \quad (3)$$

Here, λ is a regularization parameter, ξ_i is the tolerable error of the i^{th} training sample, $\|\beta\|^2$ is the parameter of the separating hyperplane and also known as the structural risk, $\|\xi\|^2$ is the sum error square, also known as the empirical risk. The structural risk maximizes the margin for separating classes [27]. Here, β is the output weight vector between the hidden and the output layer. The mathematical model described in (3) is obtained by [26] which can be defined as

$$\beta = \begin{cases} \Phi^T \left(\frac{I}{\lambda} + \Phi \Phi^T \right)^{-1} \mathbf{T} & \text{if } N < L \\ \left(\frac{I}{\lambda} + \Phi^T \Phi \right)^{-1} \Phi^T \mathbf{T} & \text{if } N > L \end{cases} \quad (4)$$

Here, I represents the identity matrix of appropriate dimension. The output for a sample $\mathbf{x}$ which comprises of m inputs is obtained as follows.

$$\mathbf{f}(\mathbf{x}) = \begin{cases} \text{sign } \phi(\mathbf{x}) \Phi^T \left(\frac{I}{\lambda} + \Phi \Phi^T \right)^{-1} \mathbf{T} & \text{if } N < L \\ \text{sign } \phi(\mathbf{x}) \left(\frac{I}{\lambda} + \Phi^T \Phi \right)^{-1} \Phi^T \mathbf{T} & \text{if } N > L \end{cases} \quad (5)$$

Here, $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \dots, f_m(\mathbf{x})]$ is the output vector. The predicted label of $\mathbf{x}$ is defined as

$$\text{label}(\mathbf{x}) = \underset{k}{\text{argmax}} f_k(\mathbf{x}), \quad k = 1, \dots, m. \quad (6)$$

2.2 Kernelized extreme learning machine

Kernelized extreme learning machine proposed in [26] has good generalization performance compared to the traditional ELM that uses random input parameters. This is due to the use of kernel function to transform the data to the kernel space. An explicit feature mapping function $\phi(x)$ is actually no longer needed in the kernelized extreme learning machine as it uses kernel trick. KELM can map the features into higher dimensional space in order to enhance the separability of the input dataset. The equation for the popular Gaussian kernel function is delineated below.

$$K(\mathbf{x}_i, \mathbf{x}_j^c) = \exp \left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j^c\|^2}{\sigma^2} \right) \quad (7)$$

Here, σ is the kernel width parameter, $\mathbf{x}_i$ represents the i^{th} sample and $\mathbf{x}_j^c$ represents the j^{th} centroid $i, j \in 1, 2, \dots, N$. The number of Gaussian kernel function utilized in [26] is same as the number of training samples. The kernel matrix of the hidden layer is given by the following Eq.

$$\Phi = \begin{bmatrix} K(x_1, x_1^c) & K(x_1, x_2^c) & \dots & K(x_1, x_N^c) \\ K(x_2, x_1^c) & K(x_2, x_2^c) & \dots & K(x_2, x_N^c) \\ \vdots & \vdots & \ddots & \vdots \\ K(x_N, x_1^c) & K(x_N, x_2^c) & \dots & K(x_N, x_N^c) \end{bmatrix}_{N \times N} \quad (8)$$

Here, $K(\mathbf{x}_i, \mathbf{x}_j^c)$ is the output of the j^{th} hidden neuron for the i^{th} input, $\mathbf{x}_i$, here $i, j \in 1, 2, \dots, N$. All the notations used in this section are same as that of sect. 2.1. In kernelized ELM the problem of finding the output layer weights is formulated as follows:

$$\begin{aligned} &\text{Minimize: } \frac{1}{2} \|\beta\|^2 + \lambda \frac{1}{2} \sum_{i=1}^N \|\xi_i\|^2 \\ &\text{Subject to: } \phi(\mathbf{x}_i) \beta = \mathbf{t}_i^T - \xi_i^T, \quad i = 1, \dots, N \end{aligned} \quad (9)$$

Using the KKT theorem the constrained optimization formulation in Eq. (9) is changed to a dual optimization problem as shown below.

$$\mathcal{L}_{DELM} = \frac{1}{2} \|\beta\|^2 + \lambda \frac{1}{2} \sum_{i=1}^N \|\xi_i\|^2 - \sum_{i=1}^N \alpha_i (\phi(\mathbf{x}_i)\beta - \mathbf{t}_i^T + \xi_i^T) \quad (10)$$

Here, $\mathcal{L}_{DELM}$ represents the dual optimization problem. α_i is the Lagrangian coefficient for the sample $\mathbf{x}_i$. For the small training samples, the solution is determined in [26] which is given below.

$$\beta = \Phi^T \left(\frac{I}{\lambda} + \Phi \Phi^T \right)^{-1} \mathbf{T} \quad (11)$$

Here, I represents the identity matrix of, $N \times N$ dimension. Applying Mercer's condition to define the kernel matrix of KELM as illustrated in the following Eq.

$$\Omega_{ELM} = \Phi \Phi^T : \Omega_{ELM_{i,j}} = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j^c) \quad (12)$$

The output can be determined by the following expression

$$\mathbf{f}(\mathbf{x}) = \text{sign} \left[\begin{matrix} K(\mathbf{x}, \mathbf{x}_1^c) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N^c) \end{matrix} \right]^T \left(\frac{I}{\lambda} + \Omega_{ELM} \right)^{-1} \mathbf{T} \quad (13)$$

2.3 Weighted extreme learning machine

WELM [11] was proposed for handling the imbalanced classification problems effectively. In WELM [11] each training sample is allocated a weight, The associated weight for the minority class samples is relatively larger compared to the majority class samples. Thus, the impact of the minority class is strengthened whereas, the relative impact of the majority class is diminished. The two weighting schemes empirically used to compute the weight matrix, $\mathbf{W}$ utilizing the class distribution are given below.

Weighting scheme W1:

$$W_{ii} = \frac{1}{q_k} \quad (14)$$

Here, q_k is the total number of samples belonging to the k^{th} class.

Weighting scheme W2:

$$W_{ii} = \begin{cases} \frac{0.618}{q_k} & \text{if } (q_k > q_{avg}) \\ \frac{1}{q_k} & \text{if } (q_k \leq q_{avg}) \end{cases} \quad (15)$$

Here, q_{avg} represents the average number of samples per class. Weight, W_{ii} is assigned to the i^{th} samples.

The mathematic model of WELM [11] can be described as

$$\begin{aligned} \text{Minimize: } & \frac{1}{2} \|\beta\|^2 + \frac{1}{2} \lambda \sum_{i=1}^N W_{ii} \|\xi_i\|^2 \\ \text{Subject to: } & \phi(\mathbf{x}_i)\beta = \mathbf{t}_i^T - \xi_i^T, \quad i = 1, \dots, N \end{aligned} \quad (16)$$

Here, a diagonal matrix $\mathbf{W} = \text{diag}(W_{ii})$ is associated to allocate weight to each training sample $\mathbf{x}_i$. Based on the Karush–Kuhn–Tucker (KKT) theorem, the solution of (16) is delineated below.

$$\beta = \begin{cases} \Phi^T \left(\frac{I}{\lambda} + \mathbf{W}\Phi\Phi^T \right)^{-1} \mathbf{W}\mathbf{T} & \text{when } N < L \\ \left(\frac{I}{\lambda} + \Phi^T\mathbf{W}\Phi \right)^{-1} \Phi^T\mathbf{W}\mathbf{T} & \text{when } N > L \end{cases} \quad (17)$$

The solution for kernelized WELM (KWELM) (16) is described in [11] can be rewritten as follows.

$$\beta = \Phi^T \left(\frac{I}{\lambda} + \mathbf{W}\Phi\Phi^T \right)^{-1} \mathbf{W}\mathbf{T} \quad (18)$$

2.4 WELM based on overall distribution (ODW-ELM)

ODW-ELM [25] minimizes the weighted least squares error to handle the class imbalance problem. ODW-ELM uses the information theory based weighting schemes which is more effective in contrast with weighting schemes of WELM, which is shown in [25]. ODW-ELM assigns a weight to each class instead of assigning weights to all the training samples to handle the class imbalance problem. ODW-ELM does not need weight matrix, which makes it computationally efficient compared to WELM. The weighting scheme of ODW-ELM is delineated below.

$$r_k = \lfloor \log_2 p_k \rfloor, \quad k = 1, 2, \dots, m \quad (19)$$

Here, m represents the number of classes and r_k is a weight corresponding to the k^{th} class. Here, p_k is the probability corresponding to the k^{th} class.

The problem formulation of ODW-ELM is reproduced below:

$$\begin{aligned} \text{Minimize: } & \frac{1}{2} \|\beta\|^2 + \frac{1}{2} \lambda \sum_{k=1}^m r_k \|\tilde{\xi}_k\|^2 \\ \text{Subject to } & \phi(\mathbf{x}_i)\beta = \mathbf{y}_i^T - \xi_i^T, \quad i = 1, \dots, N \end{aligned} \quad (20)$$

Here, $\tilde{\xi}_k$ represents the error vector of the k^{th} class and $\bigcup_i^N \xi_i = [\tilde{\xi}_k]_{k=1}^m$. The Lagrangian of the above problem as given in [25] is reproduced below.

$$\begin{aligned} \mathcal{L}_{DELM} = & \frac{1}{2} \|\beta\|^2 + \frac{1}{2} \lambda \sum_{k=1}^m r_k \|\tilde{\xi}_k\|^2 \\ & - \sum_{i=1}^N \sum_{k=1}^m \alpha_{i,k} (\phi(\mathbf{x}_i)\beta_k - \mathbf{y}_{i,k} + \xi_{i,k}) \end{aligned} \quad (21)$$

Here, $\mathcal{L}_{DELM}$ represents the dual of the optimization problem.

The solution of the problem given in (21) is determined by [25], which is reproduced below:

$$\beta = \begin{cases} \Phi^T \left(\sum_{k=1}^m \frac{I}{\lambda r_k} + \Phi \Phi^T \right)^{-1} \mathbf{T} & \text{when } N < L \\ \left(\sum_{k=1}^m \frac{I}{\lambda r_k} + \Phi^T \Phi \right)^{-1} \Phi^T \mathbf{T} & \text{when } N > L \end{cases} \quad (22)$$

Using Eqs. (12) and (22) the kernelized ODW-ELM (ODW-KELM) can be rewritten as follow:

$$\beta = \Phi^T \left(\sum_{k=1}^m \frac{I}{\lambda r_k} + \Phi \Phi^T \right)^{-1} \mathbf{T} \quad (23)$$

3 Proposed work

3.1 Class-specific WELM based on overall distribution (OD-CSELM)

The size of weight matrix of WELM has significant effect on the computational complexity of WELM. The proposed algorithm assigns weights to the class instead of assigning weight to the training samples. The proposed OD-CSELM, significantly faster compared to WELM to handle the larger class imbalance problem more effectively. The mathematical formulation of OD-CSELM is given below.

$$\begin{aligned} \text{Minimize: } & \frac{1}{2} \|\beta\|^2 + \frac{1}{2} \lambda \left(|\log_2 p_+| \|\xi_+\|^2 + |\log_2 p_-| \|\xi_-\|^2 \right) \\ \text{Subject to: } & \phi(\mathbf{x}_i) \beta = \mathbf{y}_i^T - \xi_i^T, \quad i = 1, \dots, N \end{aligned} \quad (24)$$

The Lagrangian function of (24) is given below.

$$\mathcal{L}_{DELM} = \frac{1}{2} \|\beta\|^2 + \frac{1}{2} \lambda \left(|\log_2 p_+| \|\xi_+\|^2 + |\log_2 p_-| \|\xi_-\|^2 \right) - \sum_{i=1}^N \sum_{k=1}^m \alpha_{i,k} (\phi(x_i) \beta_k - y_{i,k} + \xi_{i,j})$$

$$\begin{aligned} \mathcal{L}_{DELM} &= \frac{1}{2} \|\beta\|^2 + \frac{1}{2} \lambda \left(|\log_2 p_+| \|\xi_+\|^2 + |\log_2 p_-| \|\xi_-\|^2 \right) \\ &- \sum_{i=1}^{N_+} \sum_{k=1}^m \alpha_{i,k} (\phi(x_i) \beta_k - y_{i,k} + \xi_{i,k}) - \sum_{j=1}^{N_-} \sum_{k=1}^m \alpha_{j,k} (\phi(x_j) \beta_k - y_{j,k} + \xi_{j,k}) \\ \mathcal{L}_{DELM} &= \frac{1}{2} \|\beta\|^2 + \frac{1}{2} \lambda \left(|\log_2 p_+| \|\xi_+\|^2 + |\log_2 p_-| \|\xi_-\|^2 \right) \\ &- \alpha^+ (\Phi_+ \beta - \mathbf{T}_+ + \xi_+) - \alpha^- (\Phi_- \beta - \mathbf{T}_- + \xi_-) \end{aligned} \quad (25)$$

Here, $\mathcal{L}_{DELM}$ represents dual optimization problem. N_+ represents the number of samples belonging to minority class and N_- represents the number of samples belonging to majority class. The proposed work assigns weights to the training samples

as per weighting approach of ODW-ELM mentioned in Eq. (19). This work uses class-specific weights $\mathbf{r}_+$ and $\mathbf{r}_-$ which can be computed as follows:

$$\mathbf{r}_+ = |\log_2 p_+|, \mathbf{r}_- = |\log_2 p_-| \quad (26)$$

Here, p_+ is the probability of the majority class and p_- is the probability of the minority class respectively. And α_i is the Lagrangian coefficient for the equality constraint corresponding to the sample $\mathbf{x}_i$. Here,

$$\alpha_i = [\alpha_{i,1}, \alpha_{i,2}, \dots, \alpha_{i,m}]^T, \quad \alpha = [\alpha_1, \alpha_2, \dots, \alpha_N]^T \quad (27)$$

$$y_i = [y_{i,1}, y_{i,2}, \dots, y_{i,m}]^T, \quad \mathbf{T} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]^T \quad (28)$$

$$\phi(\mathbf{x}_i) = [\phi_1(x_i), \phi_2(x_i), \dots, \phi_L(x_i)]^T, \quad \Phi = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \dots, \phi(\mathbf{x}_N)]^T \quad (29)$$

$$\xi = \begin{bmatrix} \xi_+ \\ \xi_- \end{bmatrix}, \Phi = \begin{bmatrix} \Phi_+ \\ \Phi_- \end{bmatrix}, \alpha = \begin{bmatrix} \alpha^+ \\ \alpha^- \end{bmatrix} \text{ and } \mathbf{T} = \begin{bmatrix} \mathbf{T}_+ \\ \mathbf{T}_- \end{bmatrix} \quad (30)$$

$$\Phi_+ = [\phi(\mathbf{x}_1^+), \dots, \phi(\mathbf{x}_{N_+}^+)]^T, \quad \Phi_- = [\phi(\mathbf{x}_1^-), \dots, \phi(\mathbf{x}_{N_-}^-)]^T \quad (31)$$

$$\alpha^+ = [\alpha_1^+, \dots, \alpha_{N_+}^+]^T, \quad \alpha^- = [\alpha_1^-, \dots, \alpha_{N_-}^-]^T \quad (32)$$

$$\mathbf{T}_+ = [\mathbf{y}_1^+, \dots, \mathbf{y}_{N_+}^+]^T, \quad \mathbf{T}_- = [\mathbf{y}_1^-, \dots, \mathbf{y}_{N_-}^-]^T \quad (33)$$

Here, Φ_+ and Φ_- are the output of hidden layer corresponding to the minority class and the majority class training samples respectively. The vectors α^+ and α^- are the Lagrangian coefficient for the equality constraints corresponding to the minority class and the majority class training samples respectively. The vectors $\mathbf{T}_+$ and $\mathbf{T}_-$ are the actual output of the minority class and the majority class training samples respectively.

$$\begin{aligned} \mathcal{L}_{DELM} = & \frac{1}{2} \|\beta\|^2 + \frac{1}{2} \lambda \left(\mathbf{r}_+ \|\xi_+\|^2 + \mathbf{r}_- \|\xi_-\|^2 \right) \\ & - [\alpha^+ \quad \alpha^-] \left(\begin{bmatrix} \Phi_+ \\ \Phi_- \end{bmatrix} \beta - \begin{bmatrix} \mathbf{T}_+ \\ \mathbf{T}_- \end{bmatrix} + \begin{bmatrix} \xi_+ \\ \xi_- \end{bmatrix} \right) \end{aligned} \quad (34)$$

The solution of the above dual optimization problem can be obtained by taking partial derivatives of $\mathcal{L}_{DELM}$ with respect to the variables $(\beta, \xi_+, \xi_-, \alpha^+, \alpha^-)$ as shown in the

following Eqs.

$$\frac{\partial \mathcal{L}_{DELM}}{\partial \beta} = 0 \Rightarrow \beta = (\Phi_+^T \alpha^+ + \Phi_-^T \alpha^-) \quad (35)$$

$$\frac{\partial \mathcal{L}_{DELM}}{\partial \xi_+} = 0 \Rightarrow \lambda \mathbf{r}_+ \xi_+ - \alpha^+ = 0 \quad (36)$$

$$\frac{\partial \mathcal{L}_{DELM}}{\partial \xi_-} = 0 \Rightarrow \lambda \mathbf{r}_- \xi_- - \alpha^- = 0 \quad (37)$$

$$\frac{\partial \mathcal{L}_{DELM}}{\partial \alpha^+} = 0 \Rightarrow (\Phi_+ \beta - \mathbf{T}_+ + \xi_+) = 0 \quad (38)$$

$$\frac{\partial \mathcal{L}_{DELM}}{\partial \alpha^-} = 0 \Rightarrow (\Phi_- \beta - \mathbf{T}_- + \xi_-) = 0 \quad (39)$$

The above mentioned, Eqs. (35), (36) and (37) are expressed as

$$\beta = (\Phi_+^T \alpha^+ + \Phi_-^T \alpha^-) \quad (40)$$

$$\alpha^+ = \lambda \mathbf{r}_+ \xi_+ \quad (41)$$

$$\alpha^- = \lambda \mathbf{r}_- \xi_- \quad (42)$$

Using (41) and (42), eq. (40) is expressed as

$$\beta = (\Phi_+^T \lambda \mathbf{r}_+ \xi_+ + \Phi_-^T \lambda \mathbf{r}_- \xi_-)$$

$$\frac{\beta}{\lambda} = \Phi_+^T \mathbf{r}_+ (\mathbf{T}_+ - \Phi_+ \beta) + \Phi_-^T \mathbf{r}_- (\mathbf{T}_- - \Phi_- \beta)$$

$$\left(\frac{I}{\lambda} + \Phi_+^T \mathbf{r}_+ \Phi_+ + \Phi_-^T \mathbf{r}_- \Phi_- \right) \beta = (\Phi_+^T \mathbf{r}_+ \mathbf{T}_+ + \Phi_-^T \mathbf{r}_- \mathbf{T}_-)$$

$$\beta = \left(\frac{I}{\lambda} + \Phi_+^T \mathbf{r}_+ \Phi_+ + \Phi_-^T \mathbf{r}_- \Phi_- \right)^{-1} (\Phi_+^T \mathbf{r}_+ \mathbf{T}_+ + \Phi_-^T \mathbf{r}_- \mathbf{T}_-) \quad (43)$$

Aforementioned, Eq. determines the output weight β for the case $N > L$. Both the cases, the output weight of GCSKELM classifier is

$$\beta = \begin{cases} \left[\Phi_+^T \Phi_-^T \right] \left(\frac{I}{\lambda} + \begin{bmatrix} \mathbf{r}_+ \Phi_+ \\ \mathbf{r}_- \Phi_- \end{bmatrix} \begin{bmatrix} \Phi_+^T & \Phi_-^T \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{r}_+ \mathbf{T}_+ \\ \mathbf{r}_- \mathbf{T}_- \end{bmatrix} & \text{when } N < L \\ \left(\frac{I}{\lambda} + \begin{bmatrix} \Phi_+^T & \Phi_-^T \end{bmatrix} \begin{bmatrix} \mathbf{r}_+ \Phi_+ \\ \mathbf{r}_- \Phi_- \end{bmatrix} \right)^{-1} \begin{bmatrix} \Phi_+^T & \Phi_-^T \end{bmatrix} \begin{bmatrix} \mathbf{r}_+ \mathbf{T}_+ \\ \mathbf{r}_- \mathbf{T}_- \end{bmatrix} & \text{when } N > L \end{cases} \quad (44)$$

The output function of OD-CSELM can be determined as follows:

$$\mathbf{f}(\mathbf{x}) = \begin{cases} \text{sign } \phi(\mathbf{x}) \left[\Phi_+^T \Phi_-^T \right] \left(\frac{I}{\lambda} + \begin{bmatrix} \mathbf{r}_+ \Phi_+ \\ \mathbf{r}_- \Phi_- \end{bmatrix} \begin{bmatrix} \Phi_+^T & \Phi_-^T \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{r}_+ \mathbf{T}_+ \\ \mathbf{r}_- \mathbf{T}_- \end{bmatrix} & \text{when } N < L \\ \text{sign } \phi(\mathbf{x}) \left(\frac{I}{\lambda} + \begin{bmatrix} \Phi_+^T & \Phi_-^T \end{bmatrix} \begin{bmatrix} \mathbf{r}_+ \Phi_+ \\ \mathbf{r}_- \Phi_- \end{bmatrix} \right)^{-1} \begin{bmatrix} \Phi_+^T & \Phi_-^T \end{bmatrix} \begin{bmatrix} \mathbf{r}_+ \mathbf{T}_+ \\ \mathbf{r}_- \mathbf{T}_- \end{bmatrix} & \text{when } N > L \end{cases} \quad (45)$$

For the small training samples the equation (44) can be rewritten as follows:

$$\beta = \Phi^T \left(\frac{I}{\lambda} + \begin{bmatrix} \mathbf{r}_+ \Phi_+ \Phi_+^T & \mathbf{r}_+ \Phi_+ \Phi_-^T \\ \mathbf{r}_- \Phi_- \Phi_+^T & \mathbf{r}_- \Phi_- \Phi_-^T \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{r}_+ \mathbf{T}_+ \\ \mathbf{r}_- \mathbf{T}_- \end{bmatrix} \quad (46)$$

Algorithm 1 Proposed OD-CSELM

Input: The given input having N observations $(\mathbf{x}_i, \mathbf{y}_i)$, $i \in N$, $\mathbf{x}_i \in R^n$, $\mathbf{y}_i \in R^m$

Output: OD-CSELM model for classification.

- 1: Initialize the class-specific weight, r_+ and r_- using (26)
 - 2: Randomly select input weights, $\mathbf{U}$ and the hidden biases, $\mathbf{b}$
 - 3: Compute the hidden layer output Φ_+ and Φ_-
 - 4: Determine the output weight, β by employing (44)
 - 5: **Return:** β
-

3.2 Class-specific kernelized WELM based on overall distribution (OD-CSKELM)

The kernelized version of OD-CSKELM can be computed as follows:

$$\beta = \Phi^T \left(\frac{I}{\lambda} + \begin{bmatrix} \mathbf{r}_+ \mathbf{K}(\mathbf{x}_{N_+}, \mathbf{x}_{N_+}) & \mathbf{r}_+ \mathbf{K}(\mathbf{x}_{N_+}, \mathbf{x}_{N_-}) \\ \mathbf{r}_- \mathbf{K}(\mathbf{x}_{N_-}, \mathbf{x}_{N_+}) & \mathbf{r}_- \mathbf{K}(\mathbf{x}_{N_-}, \mathbf{x}_{N_-}) \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{r}_+ \mathbf{T}_+ \\ \mathbf{r}_- \mathbf{T}_- \end{bmatrix} \quad (47)$$

The output of OD-CSKELM can be computed using the following Eq.

$$\mathbf{f}(\mathbf{x}) = \text{sign} \left[\begin{bmatrix} K(x, x_1) \\ \vdots \\ K(x, x_N) \end{bmatrix}^T \left(\frac{I}{\lambda} + \begin{bmatrix} \mathbf{r}_+ \mathbf{K}(\mathbf{x}_{N_+}, \mathbf{x}_{N_+}) & \mathbf{r}_+ \mathbf{K}(\mathbf{x}_{N_+}, \mathbf{x}_{N_-}) \\ \mathbf{r}_- \mathbf{K}(\mathbf{x}_{N_-}, \mathbf{x}_{N_+}) & \mathbf{r}_- \mathbf{K}(\mathbf{x}_{N_-}, \mathbf{x}_{N_-}) \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{r}_+ \mathbf{T}_+ \\ \mathbf{r}_- \mathbf{T}_- \end{bmatrix} \right] \quad (48)$$

The OD-CSKELM with Gaussian kernel function maps the data from input space to feature space by using the following Eq.

$$\mathbf{K}(\mathbf{x}_{N_+}, \mathbf{x}_{N_+}) = \exp \left(\frac{-\|\mathbf{x}_{N_+} - \mathbf{x}_{N_+}\|^2}{\sigma^2} \right), \quad \mathbf{K}(\mathbf{x}_{N_+}, \mathbf{x}_{N_-}) = \exp \left(\frac{-\|\mathbf{x}_{N_+} - \mathbf{x}_{N_-}\|^2}{\sigma^2} \right)$$

$$\mathbf{K}(\mathbf{x}_{N_-}, \mathbf{x}_{N_+}) = \exp \left(\frac{-\|\mathbf{x}_{N_-} - \mathbf{x}_{N_+}\|^2}{\sigma^2} \right), \quad \mathbf{K}(\mathbf{x}_{N_-}, \mathbf{x}_{N_-}) = \exp \left(\frac{-\|\mathbf{x}_{N_-} - \mathbf{x}_{N_-}\|^2}{\sigma^2} \right)$$

Here, $\mathbf{x}_{N_+}$ and $\mathbf{x}_{N_-}$ represents the training samples belonging to the positive and the negative class respectively.

Algorithm 2 Proposed OD-CSKELM

Input: The given input having N observations $(\mathbf{x}_i, \mathbf{y}_i)$, $i \in N$, $\mathbf{x}_i \in R^n$, $\mathbf{y}_i \in R^m$

Output: OD-CSKELM model for classification.

- 1: Initialize the class-specific weight, r_+ and r_- using (26)
 - 2: Compute the hidden layer output Φ_+ and Φ_-
 - 3: Compute kernel matrix using (49)
 - 4: Determine the output weight, β by employing (47)
 - 5: **Return:** β
-

3.3 Computational cost analysis

The following subsection analyses the computational complexities of ELM, KELM, WELM, KWELM and the proposed work.

3.3.1 Computational cost of ELM

The computational cost of determining the output weight matrix, β is determined in [19] which is reproduced below.

$$O(N^3 + 2LN^2 + mLN) \quad \text{when, } N < L \quad (49)$$

$$O(L^3 + L^2N + mLN) \quad \text{when, } N > L \quad (50)$$

3.3.2 Computational cost of KELM

Kernel matrix, Ω is determined which has computational complexity of $O(nN^2)$. The computational complexity of computing inverse of kernel matrix of size $N \times N$ is equal to $O(N^3)$. The computational complexity of the output weight matrix, β given in (11) can be determined as follows:

$$O(2N^3 + nN^2 + mN^2) \quad (51)$$

3.3.3 Computational cost of WELM

The computational cost of determining the output weight matrix, β is determined in [19] which is reproduced below.

$$O(2N^3 + 3LN^2 + mLN) \quad \text{when, } N < L \quad (52)$$

$$O(L^3 + 2LN^2 + L^2N + mLN) \quad \text{when, } N > L \quad (53)$$

3.3.4 Computational cost of KWELM

Kernel matrix, Ω is determined which has computational complexity of $O(nN^2)$. The matrix, $\mathbf{W}\Omega$ is determined which has computational complexity of $O(N^3 + nN^2)$. The computational complexity of computing inverse of kernel matrix of size $N \times N$ is equal to $O(N^3)$. The computational complexity of matrix, $\mathbf{W}\mathbf{T}$ is equal to $O(mN^2)$.

The computational complexity of the output weight matrix, β given in (23) can be determined as follows.

$$O(3N^3 + nN^2 + 2mN^2) \quad (54)$$

For the large training samples, the time taken for developing WELM classifier significantly increases due to the multiplication of the weight matrix, $\mathbf{W}_{N \times N}$.

3.3.5 Computational cost of OD-CSELM

The computational complexities of the matrix multiplication $\Phi^T W^- \Phi_-$, $\Phi_+^T r_+ \Phi_+$, $W^- \Phi_- \Phi_-^T$, $r_+ \Phi_+ \Phi_+^T$, $\Phi_-^T W^- \mathbf{T}$ and $\Phi_+^T r_+ \mathbf{T}$ are equal to $O(L^2 N_-)$, $O(L^2 N_+)$, $O(LN_-^2)$, $O(LN_+^2)$, $O(mLN_-)$ and $O(mLN_+)$ respectively. Computational cost of the output weight matrix, β derived in (44) can be determined as follows:

$$O(N^3 + 2LN_-^2 + 2LN_+^2 + mL(N_- + N_+)) \text{ when, } N < L \quad (55)$$

$$\begin{aligned} &O(L^3 + L^2 N_- + L^2 N_+ + mL(N_- + N_+)) \\ &O(L^3 + L^2 N + mLN) \text{ when, } N > L \end{aligned} \quad (56)$$

3.3.6 Computational cost of OD-CSKELM

The computational complexities for evaluating the kernel terms $r_+ \mathbf{K}(\mathbf{x}_{N_+}, \mathbf{x}_{N_+})$, $r_+ \mathbf{K}(\mathbf{x}_{N_+}, \mathbf{x}_{N_-})$, $r_- \mathbf{K}(\mathbf{x}_{N_-}, \mathbf{x}_{N_+})$ and $r_- \mathbf{K}(\mathbf{x}_{N_-}, \mathbf{x}_{N_-})$ have order of $O((n+1)N_+^2)$, $O((n+1)N_+N_-)$, $O((n+1)N_-N_+)$ and $O((n+1)N_-^2)$ respectively. Computational cost of computing inverse of kernel matrix of size $N \times N$ is equal to $O(N^3)$. Computational cost of the output weight matrix, β given in (47) can be determined as follows:

$$\begin{aligned} &O(2N^3 + (n+1)(N_- + N_+)^2 + m(N_+ + N_-) + mN^2) \\ &O(2N^3 + (n+m+1)N^2 + mN) \end{aligned} \quad (57)$$

4 Experimental setting and result evaluation

4.1 Dataset description

The experiments were performed to evaluate the proposed classifier by using 31 datasets were obtained from online repositories, including UCI Machine Learning Repository [28] and the KEEL data repository [29] which are available in the 5-fold cross validation format. The imbalance ratio (IR) of the datasets varies over the datasets. The imbalance ratio(IR) is calculated as

$$\text{Binary : (IR)} = \frac{\# \text{minority samples}}{\# \text{majority samples}} \quad (58)$$

Table 1 Summary of binary imbalanced datasets

Dataset	Number of samples	Number of features	Minority (%)	Majority (%)	Imbalance ratio
abalone19	4174	8	0.77	99.23	0.0078
yeast1289vs7	947	8	3.17	96.83	0.0255
yeast4	1484	8	3.44	96.56	0.0305
yeast2vs8	482	8	4.15	95.85	0.0433
yeast1458vs7	693	8	4.33	95.67	0.0452
shuttleC2vsC4	129	9	4.65	95.35	0.0488
abalone9vs18	731	8	5.65	94.25	0.0590
pageblocks13vs4	472	10	5.93	94.07	0.0631
yeast1vs7	459	7	6.72	93.28	0.0721
shuttleC0vsC4	1829	9	6.72	93.28	0.0721
ecoli4	336	7	6.85	93.15	0.0735
glass2	214	9	8.78	91.22	0.0962
glass016vs2	192	9	8.85	91.15	0.0972
vowel0	988	13	9.01	90.99	0.0990
yeast05679vs4	528	8	9.66	90.34	0.1069
yeast2vs4	514	8	9.92	90.08	0.1101
pageblocks0	5472	10	10.23	89.77	0.1140
yeast3	1484	8	10.98	89.02	0.1233
glass6	214	9	13.55	86.45	0.1567
segment0	2308	19	14.26	85.74	0.1662
ecoli2	336	7	15.48	84.52	0.1831
newthyroid1	215	5	16.28	83.72	0.1945
newthyroid2	215	5	16.28	83.72	0.1945
vehicle0	846	18	23.52	76.48	0.3076
glass0123vs456	214	9	23.84	76.17	0.3130
haberman	306	3	27.42	73.58	0.3731
vehicle1	846	18	28.37	71.63	0.3960
vehicle2	846	18	28.37	71.63	0.3960
pima	768	8	34.84	66.16	0.5356
wisconsin	683	9	35.00	65.00	0.5380
banana	5300	2	46.23	53.77	0.5495

Here, # stands "number of". The brief description of the datasets used are listed in Table 1. Normalization is done to map the numbers within a range 1 and -1, can be expressed as follows:

$$x' = \left(\frac{x - \min_n}{\max_n - \min_n} \right) \times 2 - 1 \quad (59)$$

Here, x is the original attribute value and x' is the normalized attribute value, $\max_n$ is the maximum value of feature n and $\min_n$ is the minimum value of feature n .

4.2 Parameter settings

The proposed OD-CSELM employs hidden neurons with the Sigmoid node and OD-CSKELM map the input data to the kernel space by using the Gaussian kernel func-

(a)

(b)

Fig. 1 Display of the Friedman-aligned mean-ranks for the different algorithms. (a)G-mean (b) AUC.

Fig. 2 Display of Box-plot for the different algorithms. (a)G-mean (b) AUC.

(a)

(b)

Fig. 3 Display of mean training time (in seconds). (a) Sigmoid node (b) Gaussian kernel.

Table 2 Performance evaluation for binary problems.

Dataset	G-mean		Sigmoid node						Gaussian kernel							
	ELM		WELM		ODW-ELM		OD-CSELM		KELM		KWELM		ODW-KELM		WKSMTOTE	OD-CSKELM
	(λ, L)	Testing result(%)	(λ, L)	Testing result(%)	(λ, L)	Testing result(%)	(λ, L)	Testing result(%)	(λ, σ)	Testing result(%)	(λ, σ)	Testing result(%)	(λ, σ)	Testing result(%)	Testing result(%)	Testing result(%)
abalone19	(2 ⁴² , 990)	47.52	(2 ⁶ , 150)	77.19	(2 ²⁴ , 780)	51.12	(2 ⁸ , 30)	77.81	(2 ⁴² , 2 ⁰)	51.36	(2 ²⁴ , 2 ¹⁶)	74.47	(2 ⁵⁰ , 2 ²)	52.73	75.45	(2 ¹² , 2 ¹⁰) 81.03
abalone9vs18	(2 ⁴⁰ , 150)	75.29	(2 ³² , 20)	88.72	(2 ³⁶ , 60)	72.16	(2 ¹⁰ , 250)	87.65	(2 ¹⁸ , 2 ⁰)	72.71	(2 ²⁸ , 2 ¹⁴) 89.76	(2 ⁻⁴ , 2 ⁻¹⁴)	76.56	91.94	(2 ⁴⁰ , 2 ²⁰)	87.72
yeast3	(2 ⁴⁰ , 100)	80.75	(2 ¹⁶ , 700)	93.25	(2 ²² , 280)	81.47	(2 ³⁸ , 640)	93.14	(2 ⁴⁴ , 2 ⁸)	84.13	(2 ⁴⁸ , 2 ¹⁶)	93.11	(2 ⁴⁸ , 2 ¹⁴)	83.66	91.00	(2 ⁴ , 2 ⁰) 92.36
yeast4	(2 ⁴² , 960)	65.52	(2 ⁶ , 20)	87.29	(2 ⁵⁰ , 660)	62.98	(2 ⁻⁴ , 30)	85.01	(2 ⁻⁶ , 2 ⁻¹⁶)	70.47	(2 ² , 2 ⁻⁴)	84.98	(2 ² , 2 ⁻¹⁴)	71.66	79.00	(2 ⁻² , 2 ⁻⁴) 86.80
pageblock0	(2 ³⁴ , 830)	89.92	(2 ²² , 750)	93.21	(2 ²⁴ , 30)	90.12	(2 ⁸ , 780)	93.88	(2 ⁴ , 2 ⁻⁴)	89.62	(2 ¹⁶ , 2 ⁰)	93.61	(2 ¹⁴ , 2 ⁻²)	89.82	91.14	(2 ⁸ , 2 ⁰) 93.88
pageblock13vs4	(2 ² , 40)	97.89	(2 ²⁰ , 600)	98.07	(2 ²⁴ , 560)	98.00	(2 ² , 30)	100	(2 ² , 2 ⁻²)	97.89	(2 ²⁰ , 2 ¹⁰)	98.07	(2 ⁻¹⁸ , 2 ⁻⁸)	97.89	97.38	(2 ² , 2 ⁻²) 100
glass016vs2	(2 ³⁴ , 150)	67.78	(2 ¹⁴ , 380)	83.77	(2 ⁻⁴ , 170)	68.20	(2 ⁴ , 710)	83.97	(2 ⁵⁰ , 2 ¹⁸)	63.20	(2 ³² , 2 ⁸)	83.59	(2 ⁵⁰ , 2 ¹²)	61.53	79.00	(2 ²⁸ , 2 ⁸) 83.59
shuttleC0vsC4	(2 ¹⁴ , 10)	100	(2 ³⁸ , 10)	100	(2 ¹⁴ , 10)	100	(2 ⁻¹⁸ , 10)	100	(2 ¹⁴ , 2 ⁰)	100	(2 ²⁶ , 2 ⁰)	100	(2 ¹⁸ , 2 ⁰)	100	100	(2 ⁻¹⁸ , 2 ⁰) 100
shuttleC2vsC4	(2 ⁴⁰ , 20)	93.54	(2 ³² , 10)	100	(2 ² , 20)	94.00	(2 ⁻¹⁸ , 10)	100	(2 ⁻¹⁸ , 2 ⁻¹²)	94.14	(2 ⁻¹⁸ , 2 ⁰)	100	(2 ⁻¹⁸ , 2 ⁻¹²)	94.14	100	(2 ⁻¹⁸ , 2 ⁰) 100
yeast05679vs4	(2 ³² , 390)	64.49	(2 ⁻² , 150)	81.05	(2 ¹⁶ , 200)	64.70	(2 ⁶ , 920)	81.50	(2 ⁻⁶ , 2 ⁻¹⁶)	68.68	(2 ⁴ , 2 ⁻⁶)	82.21	(2 ¹⁰ , 2 ⁰)	69.51	81.00	(2 ⁻¹⁸ , 2 ⁰) 82.22
yeast1vs7	(2 ⁴⁰ , 960)	65.58	(2 ¹⁶ , 550)	77.26	(2 ²² , 780)	68.41	(2 ⁰ , 600)	78.22	(2 ¹⁴ , 2 ⁻⁴)	64.48	(2 ²⁰ , 2 ⁴)	77.72	(2 ¹² , 2 ⁻²)	67.88	76.00	(2 ⁰ , 2 ⁻²) 78.22
yeast1458vs7	(2 ⁴⁴ , 970)	61.07	(2 ¹⁰ , 120)	67.10	(2 ⁴⁸ , 40)	61.20	(2 ² , 240)	70.05	(2 ²⁴ , 2 ⁻⁴)	59.89	(2 ⁰ , 2 ⁻⁸)	69.32	(2 ³² , 2 ²)	57.91	67.00	(2 ⁻⁶ , 2 ⁻⁶) 71.52
yeast1289vs7	(2 ⁴⁵ , 880)	59.23	(2 ³² , 120)	75.83	(2 ⁴⁸ , 740)	65.82	(2 ⁰ , 60)	75.35	(2 ¹⁸ , 2 ¹⁰)	60.97	(2 ⁴ , 2 ⁻²)	71.41	(2 ³² , 2 ⁰)	56.69	69.83	(2 ⁰ , 2 ⁻⁴) 75.66
yeast2vs8	(2 ⁰ , 290)	72.83	(2 ⁴² , 960)	75.56	(2 ⁻² , 20)	72.83	(2 ⁴ , 200)	82.73	(2 ²⁴ , 2 ⁻²)	77.24	(2 ¹⁴ , 2 ⁻⁴)	77.89	(2 ²⁶ , 2 ⁻²)	78.91	80.00	(2 ¹² , 2 ⁰) 86.02
yeast2vs4	(2 ³⁶ , 280)	86.25	(2 ³⁶ , 940)	91.56	(2 ⁴⁶ , 20)	87.79	(2 ⁴ , 380)	92.08	(2 ⁶ , 2 ⁻⁴)	85.31	(2 ³² , 2 ⁴)	91.88	(2 ⁶ , 2 ⁻²)	85.30	90.00	(2 ⁴ , 2 ²) 92.29
ecoli4	(2 ²² , 60)	91.96	(2 ⁶ , 180)	97.83	(2 ¹² , 300)	91.96	(2 ⁸ , 40)	98.72	(2 ⁴⁶ , 2 ¹⁶)	94.32	(2 ⁶ , 2 ⁰)	98.24	(2 ² , 2 ⁻²)	89.28	92.00	(2 ⁴⁶ , 2 ¹⁸) 98.72
glass2	(2 ³⁸ , 110)	79.49	(2 ³² , 140)	80.33	(2 ⁻⁸ , 20)	80.42	(2 ³⁰ , 320)	82.24	(2 ¹⁶ , 2 ⁻²)	66.28	(2 ¹⁴ , 2 ²)	82.59	(2 ¹⁸ , 2 ⁻²)	66.75	79.00	(2 ¹⁰ , 2 ⁻²) 84.39
vowel2	(2 ³⁸ , 110)	100	(2 ³⁰ , 120)	100	(2 ⁻¹⁸ , 10)	100	(2 ⁻¹⁶ , 30)	100	(2 ⁻¹⁸ , 2 ⁻¹⁰)	100	(2 ⁻¹⁸ , 2 ⁻¹⁰)	100	(2 ⁻¹⁸ , 2 ⁻¹⁰)	100	100	(2 ⁻¹⁶ , 2 ⁻⁶) 100
haberman	(2 ⁴⁴ , 910)	49.16	(2 ³⁴ , 10)	65.11	(2 ⁴² , 280)	54.79	(2 ⁴ , 120)	66.91	(2 ⁴² , 2 ⁰)	57.23	(2 ¹⁴ , 2 ⁴)	66.26	(2 ⁻¹⁴ , 2 ⁻¹⁶)	64.78	65.21	(2 ⁴² , 2 ¹⁸) 66.91
pima	(2 ³² , 30)	70.10	(2 ¹⁴ , 20)	74.74	(2 ⁻⁴ , 90)	71.08	(2 ² , 230)	75.88	(2 ⁰ , 2 ²)	71.16	(2 ⁸ , 2 ²)	75.58	(2 ² , 2 ⁰)	71.30	74.00	(2 ⁻² , 2 ⁰) 75.94
wisconsin	(2 ³⁷ , 50)	96.32	(2 ³⁴ , 60)	97.07	(2 ³⁰ , 260)	97.18	(2 ⁻⁴ , 990)	97.55	(2 ⁻² , 2 ²)	97.22	(2 ⁸ , 2 ⁴)	97.70	(2 ¹⁸ , 2 ¹²)	97.07	96.33	(2 ⁸ , 2 ⁰) 98.00
vehicle0	(2 ⁴⁰ , 460)	98.57	(2 ¹⁶ , 850)	99.32	(2 ⁶ , 850)	99.44	(2 ⁶ , 20)	99.60	(2 ⁸ , 2 ²)	99.36	(2 ²⁰ , 2 ⁴)	99.36	(2 ⁶ , 2 ²)	99.29	99.14	(2 ¹² , 2 ⁴) 99.98
vehicle1	(2 ⁸ , 570)	79.29	(2 ¹⁴ , 450)	85.30	(2 ²⁴ , 450)	81.13	(2 ⁰ , 220)	85.40	(2 ¹⁸ , 2 ⁸)	80.60	(2 ²⁴ , 2 ⁸)	86.74	(2 ²⁸ , 2 ¹⁰)	79.64	86.00	(2 ²² , 2 ¹⁰) 91.76
vehicle2	(2 ¹² , 600)	98.43	(2 ¹⁶ , 800)	99.12	(2 ⁴⁴ , 40)	98.76	(2 ²² , 670)	99.23	(2 ⁸ , 2 ⁰)	99.12	(2 ²⁰ , 2 ²)	99.45	(2 ⁸ , 2 ⁰)	99.22	98.82	(2 ⁶ , 2 ⁰) 99.71
newthyroid1	(2 ¹⁸ , 180)	98.24	(2 ¹⁸ , 30)	99.44	(2 ¹² , 540)	98.52	(2 ⁻⁴ , 220)	99.72	(2 ⁰ , 2 ⁻⁴)	99.16	(2 ¹⁴ , 2 ⁰)	99.72	(2 ² , 2 ⁻⁴)	99.16	88.69	(2 ⁶ , 2 ⁻²) 99.72
newthyroid2	(2 ¹⁸ , 40)	95.55	(2 ¹⁶ , 290)	99.72	(2 ²⁴ , 20)	96.70	(2 ¹⁸ , 210)	99.72	(2 ² , 2 ⁻⁴)	99.44	(2 ¹² , 2 ⁻⁴)	99.72	(2 ² , 2 ⁻⁴)	99.44	90.72	(2 ⁻⁴ , 2 ⁻⁸) 99.44
glass0123vs456	(2 ⁸ , 140)	90.67	(2 ¹⁰ , 290)	94.68	(2 ²⁴ , 120)	91.60	(2 ² , 700)	95.73	(2 ¹⁰ , 2 ⁻²)	93.26	(2 ⁻¹⁸ , 2 ⁻⁴)	95.41	(2 ⁸ , 2 ⁻²)	93.26	94.19	(2 ⁻¹⁸ , 2 ⁻⁴) 95.41
ecoli2	(2 ³⁶ , 60)	91.17	(2 ³⁸ , 40)	93.91	(2 ¹⁶ , 120)	91.24	(2 ⁻⁴ , 280)	94.25	(2 ⁻¹⁸ , 2 ⁻⁶)	94.31	(2 ⁸ , 2 ⁻²)	94.43	(2 ⁻¹⁸ , 2 ⁻⁴)	94.31	95.00	(2 ⁻⁴ , 2 ⁻⁴) 94.25
glass6	(2 ⁴⁰ , 450)	91.96	(2 ³⁴ , 30)	95.72	(2 ⁶ , 940)	91.35	(2 ²⁶ , 30)	96.66	(2 ⁸ , 2 ²)	92.60	(2 ³⁰ , 2 ⁸)	94.04	(2 ⁻¹⁸ , 2 ⁻¹⁰)	92.84	90.00	(2 ²² , 2 ⁰) 96.04
segment0	(2 ⁸ , 720)	99.24	(2 ¹⁸ , 870)	99.75	(2 ⁻⁸ , 830)	99.24	(2 ⁶ , 640)	99.80	(2 ⁴ , 2 ⁻⁴)	99.57	(2 ¹² , 2 ⁰)	99.77	(2 ⁴ , 2 ⁻⁴)	99.60	100	(2 ⁶ , 2 ⁰) 99.80
banana	(2 ²⁴ , 50)	88.98	(2 ³⁴ , 50)	89.13	(2 ³⁰ , 20)	89.24	(2 ⁸ , 100)	90.28	(2 ² , 2 ⁰)	89.33	(2 ¹² , 2 ⁰)	89.26	(2 ² , 2 ⁰)	89.30	84.34	(2 ⁰ , 2 ⁻⁴) 90.62
Avg. G-mean		82.15		89.07		83.61		89.78		83.00		89.24		83.19	86.79	90.32

tion. This work reveals the optimal results obtained for OD-CSKELM by the performing a grid search for the regularization parameter, λ on $\{2^{-18}, 2^{-16}, \dots, 2^{48}, 2^{50}\}$ and the kernel parameter, σ on $\{2^{-18}, 2^{-16}, \dots, 2^{18}, 2^{20}\}$. The proposed OD-CSELM with Sigmoid node uses random weights between the input and the hidden layer so, its performance fluctuates. Therefore, this work reports the averaged performance of this classifier for 10 independent trials. This work demonstrate the optimal results obtained for OD-CSELM with the sigmoid node by performing grid search for the regularization parameter, λ on $\{2^{-18}, 2^{-16}, \dots, 2^{48}, 2^{50}\}$ and the number of the hidden neurons, L on $\{10, 20, \dots, 990, 1000\}$.

4.3 Evaluation of imbalanced datasets

Following the assessment are metrics traditionally employed to compute the classifier performance, not all are suitable when the dataset is imbalanced.

$$\text{Overall prediction accuracy} = \frac{T_P + T_N}{T_P + F_P + T_N + F_N} \quad (60)$$

Here, T_P is the number of correctly classified minority samples, T_N is the number of correctly classified majority samples, F_P is the number of falsely classified majority

Table 3 Performance evaluation for binary problems.

Dataset	AUC		Sigmoid node						Gaussian kernel									
	ELM		WELM		ODW-ELM		OD-CSELM		KELM		KWELM		ODW-KELM		WKSMTOTE		OD-CSKELM	
	(λ, L)	Testing result(%)	(λ, L)	Testing result(%)	(λ, L)	Testing result(%)	(λ, L)	Testing result(%)	(λ, σ)	Testing result(%)	(λ, σ)	Testing result(%)	(λ, σ)	Testing result(%)	(λ, σ)	Testing result(%)	(λ, σ)	Testing result(%)
abalone19	($2^{40}, 950$)	51.36	($2^4, 380$)	78.08	($2^{14}, 50$)	58.21	($2^8, 20$)	78.81	($2^{40}, 2^0$)	62.98	($2^{22}, 2^{16}$)	78.51	($2^{50}, 2^2$)	60.86	77.72	($2^8, 2^8$)	82.65	
abalone9vs18	($2^{24}, 940$)	75.93	($2^{10}, 900$)	94.91	($2^{26}, 20$)	76.47	($2^{20}, 10$)	95.12	($2^{22}, 2^2$)	78.03	($2^{46}, 2^{20}$)	94.13	($2^{48}, 2^{12}$)	80.35	90.91	($2^{14}, 2^8$)	95.20	
yeast3	($2^{16}, 140$)	81.79	($2^{24}, 160$)	92.99	($2^{22}, 280$)	82.63	($2^{38}, 30$)	92.92	($2^4, 2^{-2}$)	84.06	($2^{14}, 2^4$)	96.58	($2^{12}, 2^{-6}$)	84.76	92.00	($2^{-4}, 2^2$)	97.12	
yeast4	($2^6, 250$)	68.21	($2^3, 400$)	90.34	($2^{50}, 610$)	65.11	($2^{-4}, 600$)	91.77	($2^0, 2^{-6}$)	68.12	($2^{22}, 2^0$)	86.71	($2^{48}, 2^{12}$)	69.98	79.62	($2^0, 2^0$)	89.88	
pageblock0	($2^{30}, 890$)	96.06	($2^{18}, 980$)	94.47	($2^{14}, 60$)	96.62	($2^{20}, 20$)	94.88	($2^{16}, 2^0$)	90.30	($2^{16}, 2^0$)	95.45	($2^{14}, 2^{-2}$)	90.45	96.35	($2^8, 2^0$)	96.08	
pageblock13vs4	($2^{22}, 720$)	98.32	($2^{20}, 560$)	99.92	($2^{24}, 60$)	98.88	($2^2, 640$)	100	($2^{-18}, 2^{-8}$)	98.00	($2^8, 2^0$)	100	($2^{-18}, 2^{-8}$)	98.00	99.96	($2^2, 2^0$)	100	
glass016vs2	($2^{-18}, 10$)	80.16	($2^{-18}, 10$)	88.21	($2^{24}, 20$)	81.60	($2^4, 710$)	89.93	($2^8, 2^2$)	81.40	($2^{22}, 2^0$)	83.59	($2^8, 2^{20}$)	81.05	83.52	($2^{10}, 2^6$)	89.57	
shuttleC0vsC4	($2^{24}, 10$)	100	($2^{38}, 10$)	100	($2^2, 10$)	100	($2^{-18}, 10$)	100	($2^{14}, 2^0$)	100	($2^{26}, 2^0$)	100	($2^{18}, 2^0$)	100	100	($2^{-18}, 2^0$)	100	
shuttleC2vsC4	($2^{32}, 100$)	99.23	($2^{28}, 10$)	100	($2^{22}, 60$)	99.56	($2^{-18}, 10$)	100	($2^{50}, 2^{16}$)	95.00	($2^{-18}, 2^0$)	100	($2^{-18}, 2^{-12}$)	95.00	100	($2^{-18}, 2^0$)	100	
yeast05679vs4	($2^{12}, 130$)	83.37	($2^{-10}, 780$)	84.74	($2^{-2}, 60$)	83.70	($2^8, 240$)	86.69	($2^{18}, 2^6$)	74.54	($2^2, 2^{-4}$)	85.72	($2^{10}, 2^0$)	74.38	78.80	($2^{10}, 2^4$)	87.56	
yeast1vs7	($2^{40}, 960$)	65.58	($2^{12}, 460$)	79.43	($2^{28}, 760$)	66.30	($2^{-6}, 300$)	81.22	($2^8, 2^{-2}$)	72.15	($2^6, 2^{-4}$)	81.00	($2^{12}, 2^{-2}$)	71.56	82.89	($2^{10}, 14$)	84.44	
yeast1458vs7	($2^{24}, 900$)	63.89	($2^{21}, 720$)	69.32	($2^{30}, 30$)	65.40	($2^{22}, 100$)	70.05	($2^{26}, 2^2$)	61.46	($2^{14}, 2^8$)	70.63	($2^{38}, 2^8$)	60.47	74.91	($2^{-6}, 2^{-6}$)	72.56	
yeast1289vs7	($2^{42}, 880$)	63.24	($2^{44}, 20$)	74.20	($2^{-8}, 90$)	68.91	($2^{22}, 100$)	75.35	($2^{22}, 2^2$)	65.43	($2^4, 2^0$)	78.73	($2^{26}, 2^2$)	65.20	77.51	($2^0, 2^2$)	81.59	
yeast2vs8	($2^{-2}, 160$)	76.24	($2^{10}, 20$)	80.22	($2^{-2}, 20$)	77.47	($2^4, 60$)	82.73	($2^{18}, 2^{-2}$)	79.39	($2^{18}, 2^0$)	83.55	($2^{26}, 2^{-2}$)	79.39	85.56	($2^{-2}, 2^{10}$)	85.78	
yeast2vs4	($2^{40}, 40$)	89.12	($2^{16}, 900$)	93.76	($2^{46}, 20$)	88.30	($2^6, 180$)	96.17	($2^4, 2^{-2}$)	87.84	($2^{16}, 2^2$)	93.09	($2^6, 2^{-2}$)	86.90	90.43	($2^6, 2^4$)	96.40	
ecoli4	($2^{38}, 30$)	99.11	($2^2, 560$)	99.12	($2^{12}, 300$)	99.36	($2^{-6}, 60$)	99.48	($2^8, 2^{-4}$)	94.87	($2^6, 2^0$)	98.24	($2^2, 2^{-2}$)	90.00	96.45	($2^{46}, 2^{18}$)	99.60	
glass2	($2^{38}, 880$)	64.18	($2^{10}, 380$)	83.66	($2^{18}, 660$)	64.45	($2^{20}, 400$)	87.43	($2^{18}, 2^0$)	69.90	($2^{12}, 2^{40}$)	84.71	($2^{20}, 2^{-2}$)	69.90	88.80	($2^{10}, 2^2$)	88.65	
vowel0	($2^{38}, 100$)	100	($2^{40}, 100$)	100	($2^4, 100$)	100	($2^{-16}, 10$)	100	($2^{-18}, 2^{-10}$)	100	($2^{-18}, 2^{-10}$)	100	($2^{-18}, 2^{-10}$)	100	57.38	($2^{-16}, 2^{-6}$)	100	
haberman	($2^{36}, 20$)	58.51	($2^8, 520$)	67.63	($2^{42}, 280$)	59.13	($2^{42}, 90$)	68.91	($2^{42}, 2^0$)	60.31	($2^{50}, 2^{18}$)	67.81	($2^{44}, 2^{12}$)	61.77	67.34	($2^{22}, 2^{14}$)	69.68	
pima	($2^2, 220$)	73.60	($2^8, 540$)	78.72	($2^0, 560$)	73.84	($2^2, 750$)	81.19	($2^2, 2^2$)	73.94	($2^6, 2^0$)	78.30	($2^0, 2^2$)	73.96	79.18	($2^{-4}, 2^0$)	81.16	
wisconsin	($2^2, 260$)	97.81	($2^8, 880$)	98.44	($2^{20}, 540$)	98.18	($2^0, 770$)	98.67	($2^{16}, 2^{12}$)	98.15	($2^{20}, 2^{10}$)	98.75	($2^{18}, 2^{12}$)	98.15	98.80	($2^{-6}, 2^6$)	99.54	
vehicle0	($2^8, 220$)	99.36	($2^{20}, 440$)	99.60	($2^6, 850$)	99.71	($2^{-2}, 20$)	99.85	($2^{12}, 2^4$)	99.53	($2^{42}, 2^{12}$)	99.96	($2^6, 2^2$)	99.72	99.00	($2^{12}, 2^4$)	99.98	
vehicle1	($2^6, 850$)	86.76	($2^{12}, 800$)	89.42	($2^{46}, 800$)	87.12	($2^{20}, 620$)	90.40	($2^{18}, 2^8$)	86.52	($2^{38}, 2^8$)	90.62	($2^{28}, 2^{10}$)	86.59	85.22	($2^{22}, 2^{10}$)	91.76	
vehicle2	($2^8, 710$)	99.41	($2^{16}, 690$)	99.54	($2^{24}, 320$)	99.36	($2^{-8}, 780$)	99.63	($2^{14}, 2^4$)	99.41	($2^{22}, 2^4$)	99.69	($2^8, 2^0$)	99.50	98.90	($2^6, 2^0$)	99.71	
newthyroid1	($2^{14}, 800$)	98.47	($2^8, 740$)	100	($2^{12}, 540$)	98.57	($2^{-4}, 20$)	100	($2^{10}, 2^{-4}$)	99.60	($2^4, 2^0$)	100	($2^6, 2^{-4}$)	99.60	99.71	($2^{-2}, 2^0$)	100	
newthyroid2	($2^3, 400$)	99.60	($2^{12}, 320$)	100	($2^{-4}, 20$)	99.54	($2^{-18}, 10$)	100	($2^4, 2^{-4}$)	99.60	($2^{12}, -4$)	100	($2^6, 2^{-4}$)	99.60	99.92	($2^2, 2^0$)	100	
glass0123vs456	($2^{36}, 440$)	91.42	($2^{10}, 160$)	96.27	($2^{24}, 820$)	94.23	($2^{22}, 780$)	97.62	($2^{16}, 2^0$)	93.84	($2^8, 2^2$)	97.03	($2^8, 2^{-2}$)	93.84	98.86	($2^{-10}, 2^4$)	97.75	
ecoli2	($2^{10}, 780$)	92.23	($2^{24}, 180$)	94.94	($2^{16}, 120$)	92.05	($2^8, 800$)	95.59	($2^{-18}, 2^{-6}$)	92.03	($2^6, 2^{-4}$)	95.34	($2^{-4}, 2^{-4}$)	94.67	96.22	($2^{-6}, 2^0$)	95.81	
glass6	($2^{14}, 420$)	92.58	($2^{16}, 880$)	96.49	($2^{-12}, 580$)	92.94	($2^{14}, 240$)	97.66	($2^{20}, 2^6$)	93.33	($2^{30}, 2^8$)	95.00	($2^{46}, 2^{16}$)	93.33	92.00	($2^{-6}, 2^6$)	96.36	
segment0	($2^8, 720$)	99.78	($2^{16}, 820$)	99.83	($2^{-2}, 90$)	99.89	($2^6, 30$)	99.80	($2^6, 2^{-2}$)	99.80	($2^{10}, 2^0$)	99.85	($2^4, 2^{-4}$)	99.60	99.91	($2^6, 2^0$)	99.80	
banana	($2^{36}, 850$)	91.16	($2^{26}, 860$)	92.53	($2^4, 30$)	91.78	($2^{40}, 690$)	93.19	($2^6, 2^0$)	91.43	($2^0, 2^{-4}$)	90.47	($2^2, 2^0$)	92.11	86.40	($2^{-2}, 2^{-4}$)	93.80	
Avg. AUC		85.05		90.86		85.78		91.78		85.51		91.08		85.45		88.40		92.54

Table 4 The Wilcoxon signed-rank test results based on the G-mean. Here, R^+ is the sum of ranks for the datasets in which the first algorithm outperforms the second and R^- is the sum of ranks for the opposite.

Comparison	R^+	R^-	p -value	Hypothesis (0.05)
OD-CSKELM vs. ELM	435.0	0.0	$2.5631e^{-06}$	Rejected
OD-CSKELM vs. WELM	348.0	58.0	$9.5937e^{-04}$	Rejected
OD-CSKELM vs. ODW-ELM	417.0	18.0	$1.6045e^{-05}$	Rejected
OD-CSKELM vs. OD-CSELM	165.5	44.5	0.0239	Rejected
OD-CSKELM vs. KELM	405.0	1.0	$4.2284e^{-06}$	Rejected
OD-CSKELM vs. KWELM	280.5	44.5	0.0015	Rejected
OD-CSKELM vs. ODW-KELM	435.0	0.0	$2.5631e^{-06}$	Rejected
OD-CSKELM vs. WKSMTOTE	388.0	18.0	$2.5232e^{-05}$	Rejected

samples and F_N is the number of falsely classified minority samples.

$$R_{TP} = \frac{T_P}{T_P + F_N} \text{ and } R_{FP} = \frac{F_P}{T_N + F_P} \quad (61)$$

Table 5 The Wilcoxon signed-rank test results based on the AUC. Here, R^+ is the sum of ranks for the datasets in which the first algorithm outperforms the second and R^- is the sum of ranks for the opposite.

Comparison	R^+	R^-	p -value	Hypothesis (0.05)
OD-CSKELM vs. ELM	436.0	0.0	$2.4631e^{-06}$	Rejected
OD-CSKELM vs. WELM	328.0	23.0	$1.0742e^{-04}$	Rejected
OD-CSKELM vs. ODW-ELM	427.0	8.0	$5.8967e^{-06}$	Rejected
OD-CSKELM vs. OD-CSELM	251.0	74.0	0.0172	Rejected
OD-CSKELM vs. KELM	435.0	0.0	$2.5614e^{-06}$	Rejected
OD-CSKELM vs. KWELM	269.0	7.0	$6.7531e^{-05}$	Rejected
OD-CSKELM vs. ODW-KELM	435.0	0.0	$2.5614e^{-06}$	Rejected
OD-CSKELM vs. WKSMOTE	386.0	49.0	$2.6895e^{-04}$	Rejected

Table 6 The mean training time (MTT) (in seconds) for binary imbalanced problems.

Data set	Sigmoid node					Gaussian kernel			
	ELM	WELM	ODW-ELM	OD-CSELM	KELM	KWELM	ODW-KELM	WKSMOTE	OD-CSKELM
pageblocks0	0.7870	8.5239	0.7890	0.7750	3.6028	22.7756	3.9810	9.8182	6.9582
banana	0.7060	7.3336	0.8019	0.8264	3.4290	20.2836	3.9241	8.2405	6.3630
abalone19	0.5612	4.7047	0.5722	0.5966	1.6035	8.9437	1.7059	6.0858	3.0295
segment0	0.3543	1.8391	0.6310	0.4107	0.4950	1.6617	0.4951	2.5651	0.7802
shuttleC0vsC4	0.2898	1.3208	0.2922	0.3116	0.2052	1.1988	0.2134	1.2916	0.4387
yeast4	0.2510	0.9978	0.2705	0.2741	0.1228	0.4934	0.1265	1.1720	0.2559
yeast3	0.2438	0.9970	0.2611	0.2888	0.1293	0.4633	0.1382	1.1686	0.2477
abalone9vs18	0.1419	0.3850	0.1470	0.1824	0.0290	0.0784	0.0294	0.1833	0.0472
pima	0.1545	0.3898	0.1575	0.1616	0.0296	0.0774	0.0297	0.3317	0.0433
haberman	0.1012	0.1873	0.1028	0.1262	0.0029	0.0081	0.0042	0.1376	0.0064

Here, R_{TP} is the true positive rate and R_{FP} is the false positive rate.

$$\text{Precision} = \frac{T_P}{T_P + F_P} \text{ and } \text{Recall} = \frac{T_P}{T_P + F_N} \quad (62)$$

$$\text{G-mean} = \sqrt{R_{TP} \times (1 - R_{FP})} \quad (63)$$

Overall prediction accuracy is not an appropriate metric for imbalanced learning. For example, consider a dataset which has 92 samples belonging to the majority class and 08 samples belonging to the minority class. A classifier which predict all the samples as the majority class samples will have 92% prediction accuracy. G-mean metric is better than Overall prediction accuracy metric when the class distribution is not uniform. For the aforementioned example, G-mean will be equal to zero. The receiver operating characteristic (ROC) graph [30] computes the algorithm performance by changing the threshold value for the algorithm score to get distinct values of R_{TP} and R_{FP} . In the ROC graph, the X-axis is the R_{FP} and Y-axis is the R_{TP} . The area under the curve (AUC) [31, 32] can be employed to compute the performance of the method. The ideal point on the ROC graph would be (0, 1), for which all the

minority samples are classified correctly and no majority samples are misclassified as the minority. The bigger the AUC the better the generalization performance of the method. If the method is of hard type i.e. its predicted outcome is discrete class labels, then AUC can be defined as

$$\text{AUC} = \frac{\left(1 + \frac{T_P}{T_P + F_N} - \frac{F_P}{F_P + T_N}\right)}{2} \quad (64)$$

4.4 Experiment Results

The effectiveness of the proposed methods was investigated in term of G-mean and AUC results are listed in Table 2 and 3. The experimental results for ELM and WELM are taken from [11]. The experimental results of weighted kernel based SMOTE (WKSMOTE) [33] in term of G-mean for some of the dataset are taken from [33] and the results for remaining are obtained by experimentation. It can be observed from Tables 2 and 3 that OD-CSKELM outperforms with the rest of the classifiers for maximum number of datasets. It is also shown in Fig. 3 that OD-CSKELM outperforms ELM, WELM, ODW-ELM, OD-CSELM, KELM, KWELM, OD-KELM, and WKSMOTE. For further comparison of their performance, this work uses the Wilcoxon signed-rank test. The results of the test are shown in Table 4 and Table 5. The significance label for this test is fixed to 0.05. If the p-Value is lower than 0.05, then there is the significant difference between the two algorithms. The smaller the p-Value, the difference is more statistically significant. The mean ranking of each algorithm is additionally, obtained using the Friedman mean-ranks test [34] for the G-mean and AUC scores of the classifiers in consideration. The proposed OD-CSKELM classifier is elected as the control method, as it attains the lowest mean-ranks in the Friedman test which is shown in the Fig.1. Moreover, this work also presents Box plots for the distribution of the G-mean and AUC of these classifiers which are illustrated in the Fig. 2. The mean training time of the ELM, WELM, ODW-ELM, OD-CSELM, KELM, KWELM, ODW-KELM, WKSMOTE and OD-CSKELM are reported in Table 6. For imbalanced learning proposed OD-CSELM and OD-CSKELM take less training time compared to WELM and KWELM which can be observed from Table 6 and can also be observed from the Fig. 3.

5 Conclusion

The imbalanced classification problems are widely present in the real-world applications. However, most of the traditional classification algorithms are inclined towards the majority class as they are originally designed to address the balanced classification problems. This work proposes and evaluates OD-CSELM and kernelized version of OD-CSELM to address the class imbalance problem more effectively. OD-CSELM and OD-CSKELM assign a weight to each class instead of assigning weights to all the training samples to handle the class imbalance problem. The proposed OD-CSELM and OD-CSKELM have considerably reduced the computational cost in contrast with

WELM and KWELM respectively. The proposed algorithms are assessed by using the benchmark real-world imbalanced datasets downloaded from the KEEL dataset repository. The experimental results indicate the superiority of the proposed work in contrast with the rest of classifiers for most of the assessed datasets. In addition, the superiority of OD-CSELM and OD-CSKELM are also demonstrated by the statistical test. The future work includes the extension of OD-CSELM to address the multiclass imbalanced classification problems.

Compliance with Ethical Standards

Conflict of interests

The authors declare that they have no conflict of interest.

Informed Consent

Informed consent was obtained from all individual participants included in the study.

Human and Animal Rights

This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. He, H., Garcia, E.A.. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering* 2009;21(9):1263–1284.
2. Sarmanova, A., Albayrak, S.. Alleviating class imbalance problem in data mining. In: *2013 21st Signal Processing and Communications Applications Conference (SIU)*. 2013:1–4.
3. Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., Bing, G.. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications* 2017;73:220 – 239.
4. Wei, W., Li, J., Cao, L., Ou, Y., Chen, J.. Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web* 2013;16(4):449–475.
5. Zakaryazad, A., Duman, E.. A profit-driven artificial neural network (ann) with applications to fraud detection and direct marketing. *Neurocomputing* 2016;175:121 – 131.
6. Wang, S., Yao, X.. Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability* 2013;62(2):434–443.

7. Krawczyk, B., Galar, M., Jele, L., Herrera, F.. Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy. *Appl Soft Comput* 2016;38(C):714–726.
8. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., Herrera, F.. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 2012;42(4):463–484.
9. Liu, X.Y., Wu, J., Zhou, Z.H.. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 2009;39(2):539–550.
10. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.. Smote: Synthetic minority over-sampling technique. *J Artif Int Res* 2002;16(1):321–357.
11. Zong, W., Huang, G.B., Chen, Y.. Weighted extreme learning machine for imbalance learning. *Neurocomput* 2013;101:229–242.
12. YANG, X., SONG, Q., WANG, Y.. A weighted support vector machine for data classification. *International Journal of Pattern Recognition and Artificial Intelligence* 2007;21(05):961–976.
13. Huang, G.B., Zhu, Q.Y., Siew, C.K.. Extreme learning machine: Theory and applications. *Neurocomputing* 2006;70(1–3):489 – 501.
14. Janakiraman, V.M., Nguyen, X., Sterniak, J., Assanis, D.. Identification of the dynamic operating envelope of hcci engines using class imbalance learning. *IEEE Transactions on Neural Networks and Learning Systems* 2015;26(1):98–112.
15. Janakiraman, V.M., Nguyen, X., Assanis, D.. Stochastic gradient based extreme learning machines for stable online learning of advanced combustion engines. *Neurocomputing* 2016;177:304 – 316.
16. Li, K., Kong, X., Lu, Z., Wenyin, L., Yin, J.. Boosting weighted {ELM} for imbalanced learning. *Neurocomputing* 2014;128:15 – 21.
17. Shukla, S., Yadav, R.N.. Regularized weighted circular complex-valued extreme learning machine for imbalanced learning. *IEEE Access* 2015;3:3048–3057.
18. Xiao, W., Zhang, J., Li, Y., Zhang, S., Yang, W.. Class-specific cost regulation extreme learning machine for imbalanced classification. *Neurocomputing* 2017;261:70 – 82.
19. Raghuwanshi, B.S., Shukla, S.. Class-specific extreme learning machine for handling binary class imbalance problem. *Neural Networks* 2018;105:206 – 217.
20. Raghuwanshi, B.S., Shukla, S.. Class-specific cost-sensitive boosting weighted elm for class imbalance learning. *Memetic Computing* 2018;.
21. Raghuwanshi, B.S., Shukla, S.. Underbagging based reduced kernelized weighted extreme learning machine for class imbalance learning. *Engineering Applications of Artificial Intelligence* 2018;74:252 – 270.
22. Iosifidis, A., Gabbouj, M.. On the kernel extreme learning machine speedup. *Pattern Recognition Letters* 2015;68:205 – 210.
23. Iosifidis, A., Tefas, A., Pitas, I.. On the kernel extreme learning machine classifier. *Pattern Recognition Letters* 2015;54:11 – 17.
24. He, H., Ma, Y.. Class Imbalance Learning Methods for Support Vector Machines. Wiley-IEEE Press. ISBN 9781118646106; 2013:216.

25. Luo, X., Jiang, C., Wang, W., Xu, Y., Wang, J.H., Zhao, W.. User behavior prediction in social networks using weighted extreme learning machine with distribution optimization. *Future Generation Computer Systems* 2018;.
26. Huang, G.B., Zhou, H., Ding, X., Zhang, R.. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 2012;42(2):513–529.
27. Deng, W., Zheng, Q., Chen, L.. Regularized extreme learning machine. *IEEE Symposium on Computational Intelligence and Data Mining* 2009;:389–395.
28. Lichman, M.. UCI machine learning repository. 2013. URL: <http://archive.ics.uci.edu/ml>.
29. Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F.. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing* 2010;17(2-3):255–287.
30. Bradley, A.P.. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition* 1997;30(7):1145 – 1159.
31. Fawcett, T.. Roc graphs: Notes and practical considerations for researchers. Tech. Rep.; HP Labs, Tech. Rep. HPL-2003-4; 2003.
32. Huang, J., Ling, C.X.. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering* 2005;17(3):299–310.
33. Mathew, J., Pang, C.K., Luo, M., Leong, W.H.. Classification of imbalanced data by oversampling in kernel space of support vector machines. *IEEE Transactions on Neural Networks and Learning Systems* 2018;:1–12.
34. Demšar, J.. Statistical comparisons of classifiers over multiple data sets. *J Mach Learn Res* 2006;7:1–30.