An online weighted sequential extreme learning machine for class imbalanced data streams

Wang Li-wen¹,², Guo Wei¹,² and Yan Yi-cheng¹,²

¹College of Information Engineering, Yancheng Teachers University, Yancheng, Jiangsu, 224002, China
²College of Computer Science and technology, Nanjing Tech University, Nanjing, Jiangsu, 211816, China

*Corresponding author’s e-mail: weiguo031@163.com

Abstract. When general online classification algorithms deal with imbalanced data streams, there are always some problems, such as over fitting phenomenon caused by insufficient simple learning and instability of training model. In this paper, we introduce online sequential extreme learning machine (OSELM) as the basic theory model, and combine with the cost-sensitive strategy, then propose a cost-sensitive learning based online sequential extreme learning machine algorithm (C-OSELM). Firstly, in order to solve the problem that minority classes are easily misclassified due to class imbalance, use cost-sensitive strategy, by assigning different penalty parameters to various samples, a weighting matrix is constructed to improve the misclassification cost, thereby effectively alleviating the excessive deviation of decision surface. On this basis, in order to solve the problem that the penalty parameter is too single and the algorithm is not universal, the cost adjustment function is introduced to optimize the weight parameters to select the appropriate weight. Finally, 16 class II imbalanced datasets are used for comparison and verification. The experimental results show that the classification performances of the proposed C-OSELM algorithm are better than other comparative algorithms.

1. Introduction

Class imbalanced data classification is widely used in our real life, such as credit card fraud detection, network security detection, medical diagnosis section [1-4], and so on. In the class imbalance data classification problem, a few class samples often hide more valuable information, and this kind of samples often can not be fully learned because of the small number, which makes the traditional classifier more prone to bias to the majority class, resulting in the deviation of the decision surface, and can not guarantee the recognition accuracy of the minority class samples. In order to solve the problem that a few class samples are easy to be misclassified, scholars have improved the traditional classification algorithm and proposed a series of classification methods for class imbalanced data, such as Guo et al. [5] proposed the kernel support vector machine algorithm, Ping et al. [6] proposed the cost sensitive random forest classification algorithm, and other scholars proposed the improved extreme learning machine (ELM) algorithm [7-10].

Although the above algorithms can effectively alleviate the problem of misclassification of class imbalanced data, they are all offline batch learning of data sets, which is not suitable for the online classification of data streams widely existing in practical applications. In order to deal with these continuous, fast and real-time data streams, online sequential extreme learning machine (OSELM) [11]
came into being. This algorithm not only continues the advantages of ELM [7] such as fast training speed and strong learning ability, but also can learn the latest data samples online in an incremental way, instead of learning the existing historical data repeatedly. It is an excellent online learning algorithm, and has been successfully applied in the field of real-time data stream online mining [12]. In order to solve the problem of class imbalance data stream classification in online form, scholars have proposed a series of improved methods based on OSELM model. Mao et al. [13] proposed an OSELM algorithm based on two-stage hybrid strategy, and Ding et al. [14] proposed a weighted sequential online learning machine algorithm with kernel by combining kernel mapping method. In order to overcome the limitation that most algorithms are only applicable to specific scale data sets, Shukla [15] proposed a new online sequence class specific ELM algorithm. By using the regularization method of specific class, the algorithm can be applied to different scale data sets, and the computational complexity is relatively reduced. In order to further solve the problem of misclassification and improve the classification accuracy, Mirza et al. [16] proposed weighted OSELM algorithm (WOS-ELM), which uses weighted method to increase or weaken the influence of different categories of samples. Although the above methods improve the classification accuracy by adopting different strategies, and solve the classification problem of class imbalance data stream to a certain extent, these methods still have some shortcomings, such as too single parameter settings and not strong universality, its classification performance also needs to be further improved.

In this paper, we propose a cost-sensitive learning based new online sequential extreme learning machine algorithm (C-OSELM) for the classification of class imbalanced data streams. Firstly, OSELM is used as the basic model, combined with cost-sensitive learning strategy, and this algorithm adjusts the weight of different class samples by continuously optimizing the misclassification cost to reduce the misclassification rate of minority class samples, and the classification accuracy is improved. In order to verify the effectiveness of the proposed algorithm, this paper uses 16 imbalanced datasets of different sizes for experiments, and compares them with other online algorithms. The experimental results show that C-OSELM has the best classification performance.

2. OSELM algorithm

The ELM [7] is the theoretical basis of OSELM, which is a new learning method for single-hidden layer feedforward neural networks (Single-hidden Layer Feedforward Neural Network, SLFNs). Different from traditional neural network learning algorithms, ELM randomly generates input weights and hidden layer parameters, and uses the least square method to directly calculate the output weights. The ELM algorithm is briefly described as follows.

Suppose there are $N$ training samples $(x_j, t_j) \in R^n \times R^m$, where: $x_j$ is an $n \times 1$ dimensional input vector, $t_j$ is $m \times 1$ dimensional target vector. If the neural network contains $N$ hidden layer nodes, ELM can be expressed as:

$$f_N(x_j) = \sum_{i=1}^{N} \beta_i G(a_i, b_i, x_j) = t_j, \quad j = 1, 2, \ldots, N.$$  \hfill (1)

Where: $G(a_i, b_i, x_j)$ is the activation function of the $i$-th hidden node for the input vector $x_j$, $a_i$ and $b_i$ are randomly generated weight vectors and basis connecting the input node and the $i$-th hidden node, and $\beta_i \in R^m$ is the weight vector connecting the output node and the $i$-th hidden node. Formula (1) can be written as the following matrix form:

$$T = H\beta$$  \hfill (2)
\[ H = \begin{bmatrix} G(a_i, b_i, x_i) & \cdots & G(a_N, b_N, x_N) \\ \vdots & \ddots & \vdots \\ G(a_i, b_i, x_N) & \cdots & G(a_N, b_N, x_N) \end{bmatrix}_{N \times N} \]  

(3)

Where \( H \) is the hidden layer output matrix, \( \beta \) is the output weighting matrix, and \( T \) is the target matrix. Therefore, the training problem of ELM is equivalently converted to the problem of solving linear equations, and the output weight matrix \( \beta \) can be directly calculated by the least square method:

\[ \beta = (H^T H)^{-1} H^T T \]  

(4)

Among them, where \( H^T \) is the Moore-Penrose pseudo-inverse of the hidden-layer output matrix \( H \). If \( (H^T H) \) is nonsingular, it can be expressed by orthogonal projection to obtain formula (4).

In order to deal with the needs of streaming data, Liang et al. [11] proposed an online ELM algorithm (OSELM) which can incrementally learn data samples based on the original ELM algorithm. The algorithm is divided into two phases: the initialization phase and the online learning phase. The specific process is as follows:

**Step I: Initialization phase:**

1. Select part of the data set \( D_0 = \{(x_i, t_i), i = 1, 2, \cdots, N_0\} \) from the given data set \( D = \{(x_i, t_i), i = 1, 2, \cdots, N\} \) for initial training, where \( N_0 \geq \tilde{N} \).
2. Compute the initial hidden-layer output matrix \( H_0 \) according to the randomly generated hidden parameters \( a_i \) and \( b_i \) (\( i = 1, 2, \cdots, \tilde{N} \)).
3. Calculate the initial output weight \( \beta^0 = P_\theta H_0^T T_\theta \) according to formula (4), where \( P_\theta = (H_\theta^T H_\theta)^{-1} \).

**Step II: Sequential learning phase:**

Assuming that a new data chunk \( D_{k+1} = \{(x_i, t_i) | i = \left( \sum_{l=0}^{k} N_l \right) + 1, \cdots, \sum_{l=0}^{k+1} N_l \} \) arrives, \( N_{k+1} \) represents the number of samples in the \((k+1)\)th data chunk, first calculate the corresponding partial hidden-layer output matrix \( H_{k+1} \), and then update the output weighting matrix \( \beta^{k+1} \):

\[ P_{k+1} = P_k - P_k H_{k+1}^T (I + H_{k+1} P_k H_{k+1}^T)^{-1} H_{k+1} P_k \]  

(5)

\[ \beta^{k+1} = \beta^k + P_{k+1} H_{k+1}^T (T_{k+1} - H_{k+1} \beta^k) \]  

(6)

Note that the value of \( N_i \) can be 1. If \( N_i = 1 \), learning one by one can be considered as a special case of learning chunk by chunk.

3. **Proposed C-OSELM**

At present, there are three main categories of algorithms to solve the problem of class imbalance: sampling algorithms, cost-sensitive learning and ensemble algorithms. The sampling algorithm is divided into over-sampling and under-sampling [17]. The essence of these two methods is to make the number of positive and negative samples close first, and then perform classification learning. Although this method can alleviate the problem of misclassification caused by uneven data distribution to a certain extent, it is prone to overfitting, so it has certain limitations. Cost-sensitive
learning [18] is an effective method to solve the problem of class imbalance. The main idea is to define the misclassification cost and introduce the misclassification cost (weighting) matrix, so that the loss caused by the misclassification of minority classes should be greater than the loss of the correct classification, so as to reduce the misclassification rate of minority samples. Compared with other methods, the advantage of cost-sensitive learning is that there is no need to increase training complexity and time cost, and it will not cause overfitting. Based on the above analysis, in order to better solve the problem of class imbalanced data stream classification, this paper proposes a new cost-sensitive learning based online weighted sequential extreme learning machine algorithm (C-OSELM). By assigning different misclassification cost coefficients to each sample, constructing the weighting matrix, and combining the cost adjustment function to optimize its weights, effectively alleviate the problem of excessive deviation of the decision surface, reduce misclassifications, and improve classification accuracy.

3.1. Weighted strategy
The key to cost-sensitive learning is the setting of sample weights. The usual practice is to set weights based on information such as the proportions of various samples or the confusion matrix of the classification results. Here, we refer to the construction method of the weighting matrix in [19], and apply the penalty parameter (weight) set by it in the C-OSELM algorithm, and its value is:

$$W_i = 1 / \#(t_i)$$  \hspace{1cm} (7)

Where $W_i$ represents the penalty parameter corresponding to each training sample, and $\#(t_i)$ is the number corresponding to the $t_i$ type sample. Taking the two-class sample as an example, it is easy to see that the minority class (negative class) sample gets a larger penalty parameter than the majority class (positive class) sample, and when the class imbalance ratio is higher, the minority class will get a higher weight. In addition, since the weights adapted to each different data set are not uniform and changeable, they have their corresponding optimal penalty parameters. Therefore, in the classification process of the C-OSELM algorithm, the settings of the penalty parameters are not fixed. Instead, it optimizes and adjusts its weight parameters by introducing a cost adjustment function. The key adjustment strategies are as follows:

Firstly, we need to set the evaluation criteria of the algorithm. It is worth noting that in the imbalanced data classification problem, the overall classification accuracy cannot reflect the real performance of the classifier, so the commonly used G-mean value is used to evaluate the effectiveness of the classifier. The expression is as follows:

$$G_{\text{mean}} = \sqrt{TPR \times TNR} = \frac{tp \times tn}{\sqrt{tp + fn \times tn + fp}}$$  \hspace{1cm} (8)

The higher the G-mean value is, the better the classification effect is. It can be seen from equation (8) that the G-mean value depends on the true positive rate TPR and the negative rate accuracy TNR. In order to obtain the highest classification accuracy, it is required that the TPR and the TNR both reach the maximum and the two are equal or the difference can not be too large.

In related research [20-21], the G-mean value is optimized by an approximate re-adjustment method, the essence of which is to set the value of the weight within a certain range for corresponding adjustment. Inspired by it, C-OSELM maximizes the G-mean value by setting different weights in different data chunks. In order to control the calculation cost, the parameter adjustment setting in reference [21] selects one penalty parameter of positive or negative class to remain unchanged when adjusting the suboptimal weight, and the weight of the other class is adjusted to $1.1w, 1.3w, 1.5w$ or $0.6w, 0.7w, 0.8w$ to increase or decrease the TPR and TNR accordingly. The analysis shows that the above weight adjustment is suitable for most datasets, and the adjustment range will not make the decision surface deviate too much, resulting in excessive fluctuations in the results. Note that the weight of each data chunk needs to judge whether it needs to adjust the weight by observing the TPR.
and TNR obtained by using the verification set, and select the weight parameter corresponding to the maximum G-mean value to calculate the final output weighting matrix $\beta$.

It can be seen from equation (8) that when the TPR value is equal to the TNR value, the G-mean value reaches the maximum, but it is usually difficult to reach the same value. Therefore, it is necessary to ensure that the difference between the TPR and the TNR is not too much, so it needs to be calculated by the verification set to determine whether the difference is too large, the weight needs to be adjusted to obtain a higher G-mean value. In an ideal state, when the TPR and TNR are equal, the original weight remains unchanged. In the actual process, in order to reduce the operation cost, if the difference between the two values is small, no further adjustment is required, so the maximum difference is set to 0.01. If the difference between the TPR and the TNR is less than 0.01, the weight remains unchanged, otherwise the next step of weight adjustment is required. According to the above analysis, if the TPR is greater than the TNR, you need to increase the penalty parameter of negative samples or reduce the penalty parameter of positive samples to increase the TNR. Similarly, if the TNR is greater than the TPR, it is necessary to increase the penalty parameters of positive samples or reduce the penalty parameters of negative samples to increase the TPR.

Combining the above ideas into OSELM, the C-OSELM algorithm is obtained. The learning process is as follows:

1) Initialization phase:
   Firstly, we select the initial data chunk $D_0$, where the number of samples is $N_0$, $m_0^-$ is the number of negative samples in the initial chunk, and $m_0^+$ is the number of positive samples in the initial chunk. First calculate the penalty parameters of the positive and negative samples in the initial chunk, where the penalty parameters of the negative and positive samples are set to $w_0^- = \frac{1}{m_0^+}$ and $w_0^+ = \frac{1}{m_0^-}$ respectively. By adding the weighting matrix, the output matrix $H$ and the output weight matrix $B$ of the initial stage are calculated according to the above equations (3) and (4), which are expressed as:

\[ \beta_0^* = K_0^{\rightarrow 1} H_0^T W_0^T \theta \]
\[ K_0 = H_0^T W_0 H_0 \]

Where $W_0$ is the $N_0$ order diagonal matrix formed by the penalty parameters of the corresponding initial stage training samples, $W_0 = diag\left(\left\{w_0, \cdots, w_0\right\}\right) \in R^{(N_0 \times N_0)}$, and $w_0 \in \left\{w_0^-, w_0^+\right\}$. According to the results obtained in equations (9) and (10) are used to calculate the G-mean value of the validation set to determine whether it needs to adjust the weight, if necessary, use the suboptimal weight to adjust, the specific formula is as follows:

\[ \text{If } TNR > TPR, w_0 \rightarrow 1.1w_0^+, 1.3w_0^+, 1.5w_0^+ \text{ or } w_0 \rightarrow 0.6w_0^-, 0.7w_0^-, 0.8w_0^- \]
\[ \text{If } TNR < TPR, w_0 \rightarrow 0.6w_0^+, 0.7w_0^+, 0.8w_0^+ \text{ or } w_0 \rightarrow 1.1w_0^-, 1.3w_0^-, 1.5w_0^- \]

The parameters corresponding to the maximum G-mean value obtained from the verification set are selected as the output weighting matrix of the initial chunk, and the output weighting matrix $\beta_0$ is recalculated.

2) Sequential learning phase:
   When the subsequent data chunk arrives, its parameter matrix will also change accordingly. According to the least squares solution formula:

\[ \beta^1 = K_i^{\rightarrow 1} \begin{bmatrix} H_i^T & W_0^T & 0 \\ H_i^T & 0 & W_i^T \end{bmatrix} T_i \]
\[ K_i = \begin{bmatrix} H_i^T & W_0^T & 0 \\ H_i^T & 0 & W_i^T \end{bmatrix} \begin{bmatrix} H_i \\ 0 \end{bmatrix} \]

Where $w_i^+$ is the $N_i$ order diagonal matrix formed by the penalty parameters of the corresponding initial stage training samples, $w_i \in \left\{w_i^-, w_i^+\right\}$.
Among them, $H_1$, $T_1$, $W_1$ corresponds to the second data chunk $D_1$, and the corresponding penalty parameters in the diagonal matrix $W_1$ become $w_0^{(1)} = \frac{1}{m^+_0 + m^-_0}$ and $w_0^{(2)} = \frac{1}{m^+_0 + m^-_0}$, where $m^+_0$ and $m^-_0$ represent the number of positive and negative samples in the second sequence chunk. It can be seen that with the arrival of the new sequence chunk, the corresponding weights of the positive and negative samples must be changed accordingly, and the final calculation can be derived:

$$K_i = H_i^T W_i H_i + H_i^T W_i T_i = K_0 + H_i^T W_i (T_i - H_i \beta^0)$$  \hspace{1cm} (14)

$$\beta^i = \beta^0 + K_i H_i^T W_i \left( T_i - H_i \beta^0 \right)$$  \hspace{1cm} (15)

With the arrival of the $(k+1)$th data chunk, combining equations (5) and (6), the corresponding output weight is obtained:

$$K_{k+1} = K_k + H_{k+1}^T W_{k+1} H_{k+1}$$  \hspace{1cm} (16)

$$\beta^{k+1} = \beta^k + K_{k+1} H_{k+1}^T \left( T_{k+1} - H_{k+1} \beta^k \right)$$  \hspace{1cm} (17)

The corresponding $N_{k+1}$ (the number of samples of the $(k+1)$th sequence chunk) order diagonal weighting matrix can be expressed as:

$$W_{k+1} = \text{diag} \left( \left[ w_{k+1}^{1} \cdots w_{k+1}^{N_{k+1}} \right] \right) \in \mathbb{R}^{N_{k+1} \times N_{k+1}}$$  \hspace{1cm} (18)

Among them, $w_{k+1}^{1} \in \left\{ w_{k+1}^{1}, w_{k+1}^{N_{k+1}} \right\}$, $w_{k+1}^{1}$ and $w_{k+1}^{N_{k+1}}$ are respectively expressed as the penalty parameters of the positive and negative samples in the $(k+1)$th sequence chunk, and their values are:

$$w_{k+1}^{1} = \frac{1}{\sum_{j=0}^{N_{k+1}} m^+_j}, \quad w_{k+1}^{N_{k+1}} = \frac{1}{\sum_{j=0}^{N_{k+1}} m^-_j}$$  \hspace{1cm} (19)

Note that the weight setting of the data chunk in the online phase is the same as the initial phase. It is also necessary to determine whether the penalty parameter needs to be adjusted by the TPR and the TNR which are obtained from the verification set to determine the final output weighting matrix. The specific formula refers to (20).

if $TNR > TPR$, $w_{k+1}^{1} \rightarrow 1.1w_{k+1}^{1}, 1.3w_{k+1}^{1}, 1.5w_{k+1}^{1}$ or $w_{k+1}^{N_{k+1}} \rightarrow 0.6w_{k+1}^{N_{k+1}}, 0.7w_{k+1}^{N_{k+1}}, 0.8w_{k+1}^{N_{k+1}}$

if $TNR < TPR$, $w_{k+1}^{1} \rightarrow 0.6w_{k+1}^{1}, 0.7w_{k+1}^{1}, 0.8w_{k+1}^{1}$ or $w_{k+1}^{N_{k+1}} \rightarrow 1.1w_{k+1}^{N_{k+1}}, 1.3w_{k+1}^{N_{k+1}}, 1.5w_{k+1}^{N_{k+1}}$  \hspace{1cm} (20)

Select the weight corresponding to the maximum G-mean value obtained in the verification set as the output weighting matrix of the data chunk in the online phase, and calculate the output weighting matrix $\beta_{k+1}$ of the data chunk according to (17), and with the arrival of the new data chunk perform the next verification output to determine the final $N_{k+1}$ order diagonal weighting matrix $W_{k+1}$, and calculate the output weighting matrix $\beta_{k+1}$.

### 3.2. Algorithm description

**Input:** Dataset $\{x_i, t_i\}, \ x_i \in \mathbb{R}^d, \ t_i \in \mathbb{R}^r, i = 1, \ldots, N$.

**Output:** $\beta^{k+1}$

**Initialization phase:**

1. Select initial set $D_0 = \{(x_i, t_i), x_i \in \mathbb{R}^d, \ t_i \in \mathbb{R}^r, \ i = 1, 2, \ldots, N_0\}$, set hidden nodes $N \left( N_0 \geq \tilde{N} \right)$, random hidden node parameters $a_j, b_j, j = 1, \ldots, \tilde{N}$.

2. Calculate initial hidden layer output matrix:
\[
H_\theta = \begin{bmatrix}
G(a_1,b_1,x_1) & \cdots & G(a_N,b_N,x_1) \\
\vdots & \ddots & \vdots \\
G(a_1,b_1,x_N) & \cdots & G(a_N,b_N,x_N)
\end{bmatrix} (N_0 \geq \bar{N})
\]

3) Construct the initial diagonal weighting matrix: \( W_\theta = \text{diag}(\{w_0, \ldots, w_0\}) \in R^{(N_0 \times N_0)} \), where \( w_0 = \frac{1}{m_0^*} \), \( m_0^* \in \{w_0, w_0^*\} \).

4) Use equations (9) and (10) to calculate the initial weighting matrices \( K_\theta \) and \( \beta_\theta \).

5) Substitute the result obtained in 4) into the verification set to calculate its G-mean value and TNR, TPR. If \( TNR = TPR \) or \( |\text{TNR} - \text{TPR}| \leq 1\% \), the penalty parameter remains unchanged. If the condition is not satisfied, the weight adjustment is performed according to formula (11), and the final initial stage output matrices \( K_\theta \) and \( \beta_\theta \) are calculated again according to equations (9) and (10).

Sequential learning phase:

1) The \((k+1)\)th data chunk arrives: \( D_{k+1} = \{(x_i,t_i)|i = \{\sum_{l=0}^{k}N_l\} + 1, L \sum_{l=0}^{k}N_l\} \)

2) Calculate initial hidden layer output matrix: \( H_{k+1} \)

3) Calculate its diagonal weighting matrix \( W_{k+1} \) according to equations (18) and (19):

\[
W_{k+1} = \text{diag}(\{w_{k+1}, \ldots, w_{k+1}\}) \in R^{(\sum_{l=0}^{k}N_l \times \sum_{l=0}^{k}N_l)}
\]

4) Calculate new weighting matrix \( \beta_{k+1} \) according to equation (15).

5) Substitute the result obtained in 4) into the verification set to calculate its G-mean value and TNR, TPR. If \( TNR = TPR \) or \( |\text{TNR} - \text{TPR}| \leq 1\% \), the penalty parameter remains unchanged. If the condition is not satisfied, the weight adjustment is performed according to formula (20). Then the next data chunk is calculated and verified, and the final output matrix \( \beta_{k+1} \) is calculated according to equation (17).

4. Experiment and analysis results

4.1. Datasets

In order to verify the effectiveness of the proposed algorithm, this paper uses 16 class II imbalanced datasets collected from the KEEL database for simulation experiments, and compares the experimental results with OSEL-M and WOS-ELM. Each dataset is divided into three parts: training set, validation set and test set with roughly the same imbalance rate. The specific information is shown in Table 1:

| dataset                 | No. of attributes | No. of training samples | No. of testing samples | No. of validation samples | Imbalance ratio |
|-------------------------|-------------------|-------------------------|------------------------|--------------------------|-----------------|
| Pima                    | 8                 | 568                     | 100                    | 100                      | 1.90            |
| Yeast1                  | 8                 | 1034                    | 150                    | 300                      | 2.46            |
| Haberman                | 3                 | 200                     | 40                     | 66                       | 2.78            |
| Vehicle0                | 18                | 546                     | 150                    | 150                      | 3.25            |
| Segment0                | 19                | 1508                    | 300                    | 500                      | 6.02            |
| Yeast3                  | 8                 | 1034                    | 150                    | 300                      | 8.11            |
| Ecoli3                  | 7                 | 206                     | 50                     | 80                       | 8.6             |
| page-blocks0            | 10                | 3472                    | 1000                   | 1000                     | 8.79            |
| Yeast-0-2-5-7-9-ys-3-6-8| 8                 | 704                     | 100                    | 200                      | 9.14            |
| Vowel0                  | 13                | 638                     | 150                    | 200                      | 10.10           |

Table 1 Descriptions of experimental datasets
4.2. Experimental Settings
The experimental environment is PC (i7-10510 CPU @ 1.80 GHz, 8.00 GB RAM), the operating system is windows10, and the simulation software is Matlab R2017a. In the experiment, all data are normalized to the interval [-1,1]. In view of the different number of samples in the 16 datasets, they are divided into three sizes: large, medium and small. According to the algorithm setting, the number of hidden layer nodes needs to be $N < N_0$, and the value of the initial chunk $N_0$ of each dataset is set to twice the number of hidden layer nodes, and the size of the sequence chunk is determined by the size of the dataset. In order to reflect the sufficiency of the experiment and eliminate the contingency, two different parameters are set for each dataset for verification. For example, the number of hidden nodes $N$ in the small dataset is 25, 50, the medium dataset is 50, 100 and the large dataset is 100, 200. In order to verify the effectiveness of the C-OSELM algorithm, 20 experiments were performed under the same parameter conditions for each data set, and the average of the 20 experimental results was taken as the final result and filled in Table 2. Since the overall accuracy cannot reflect the effectiveness of the classifier on imbalanced data, this paper uses the geometric mean G-mean value commonly used in class imbalance problems as the evaluation criterion.

| Dataset                | No. of hidden neurons | Initializati on set size | Chunk size | OSELM | WOS-ELM | C-OSELM |
|------------------------|-----------------------|--------------------------|------------|-------|---------|---------|
| Haberman               | 25                    | 50                       | 30         | 0.4418| 0.4852  | 0.5053  |
| Led7digit-0-2-4-5-6-7-8-9-vs-1 | 25                  | 50                       | 30         | 0.6168| 0.6831  | 0.6857  |
| Shuttle-c0-vs-c4       | 25                    | 50                       | 30         | 0.7318| 0.7419  | 0.7796  |
| page-blocks-1-3-vs-4   | 25                    | 50                       | 30         | 0.4315| 0.7823  | 0.8452  |
| Abalone9-18            | 25                    | 50                       | 30         | 0.2329| 0.6507  | 0.6789  |
| Yeast-1-4-5-8-vs-7     | 25                    | 50                       | 30         | 0.2329| 0.6507  | 0.6789  |
| Yeast5                 | 25                    | 50                       | 30         | 0.2329| 0.6507  | 0.6789  |
| Yeast6                 | 25                    | 50                       | 30         | 0.2329| 0.6507  | 0.6789  |

Table 2: Number of hidden nodes, sizes of initial sets and chunks and G-mean values
4.3. Comparative analysis of experimental results
Table 2 shows the comparison results of the three algorithms of OSELM, WOS-ELM and C-OSELM. The higher the G-mean value, the more accurate the prediction results. The best value of each group of experimental results is expressed in bold. It can be seen from the experimental result:

Through comparison, it is found that the classification effect of the original OSELM algorithm is the worst, while WOS-ELM increases the misclassification cost of the sample by setting corresponding penalty parameters for the sample, so that the classification accuracy is improved. The classification results of most datasets of the C-OSELM algorithm proposed in this paper are significantly better than these two comparison algorithms. This is because the algorithm not only gives various sampled misclassification cost coefficients and forms a weighting matrix, but also takes into account the different datasets. There is a corresponding adaptation weight, that is, the C-OSELM algorithm adds a cost adjustment function on the basis of setting the penalty parameters, and optimizes its weight parameters, and the setting of the adjustment parameter range can basically cover the position of the optimal decision surface, also it will not cause excessive fluctuations in the decision surface during the parameter adjustment process, and it can also be in the best classification position of each dataset. Therefore, the C-OSELM algorithm is more effective in solving the classification problem of uneven data streams than the other two algorithms.

Two different parameters are set for each dataset, and the results show that the number of hidden layer nodes and the initial chunk size will affect the experimental results. If the number of hidden layer nodes is increased in most cases, the classification accuracy will be improved. This result verifies that increasing the number of hidden layer nodes can effectively improve the learning performance of extreme learning machine, but it does not mean that the larger the number of hidden layer nodes, the better the classification result. For example, observing the datasets page-blocks-1-3-vs-4 and yeast6 show that with the increase of the number of hidden layer nodes, the classification result decreases, indicating that each dataset will have its optimal parameter value, and the optimal classification result cannot be obtained below or above this value.

In conclusion, cost-sensitive learning can be used as an effective means to solve the class imbalance problem, and C-OSELM algorithm is active and effective to solve the classification problem of imbalanced data streams with different scales and imbalance rates.

5. Conclusion
Aiming at the problem of imbalanced data stream classification, this paper proposes an online weighted sequential extreme learning machine algorithm C-OSELM based on OSELM and cost-sensitive learning strategy. The algorithm sets different penalty parameters for each class of samples to improve the cost of misclassification, so as to effectively alleviate the situation of excessive deviation of decision surface and misclassification of minority samples, and combined with the cost adjustment function to continuously optimize and adjust the weight of various kinds of samples, the optimal weight of the dataset is selected to make the algorithm more universal. This paper is based on the assumption that there is no concept drift in the data stream. The next research needs to consider the classification learning of class imbalanced data stream with concept drift.

References
[1] Zhou P Y, Wong Andrew K C. (2021) Explanation and prediction of clinical data with imbalanced class distribution based on pattern discovery and disentanglement. BMC medical informatics and decision making, 21(1):16-16.
[2] Mardiansyah H, Widia S R, Efendi S. (2021) Handling Problems of Credit Data for Imbalanced Classes using SMOTEXGBoost. Journal of Physics: Conference Series, 1830(1).
[3] GAO T, HAO Y G, ZHANG H P, et al. (2018) Predicting Pathological Response to Neoadjuvant Chemotherapy in Breast Cancer Patients Based on Imbalanced Clinical Data. Personaland Ubiquitous Computing, 22: 1039-1047.
[4] Bhagat S R, Shukla S.(2021) Classifying multiclass imbalanced data using generalized class-
specific extreme learning machine. Progress in Artificial Intelligence, 1-23.

[5] GUO T, WANG J, LIU Q M, et al. Kernel SVM Algorithm Based on Identifying Key Samples for Imbalanced Data. Pattern Recognition and Artificial Intelligence, 2019, 32 (6): 569-576.

[6] PING R, ZHOU S S, LI D.(2020) Cost Sensitive Random Forest Classification Algorithm for Highly Unbalanced Data. Pattern Recognition and Artificial Intelligence, 33(3): 249-257.

[7] Huang G B, Zhu Q, Siew C K. (2006) Extreme learning machine: Theory and applications. Neurocomputing, 70 (1): 489-501.

[8] Nayak D R, Das D, Dash R, et al. (2020) Deep extreme learning machine with leaky rectified liner unit for multiclass classification of pathological brain images. Multi-media Tools and Applications, 79 (21): 15381-15396.

[9] NEMISSI M, SALAH H, SERIDI H. (2018) Breast cancer diagnosis using an enhanced Extreme Learning Machine based-Neural Network// International Conference on Signal, Image, Vision and their Applications (SIVA). IEEE, pp.1-4.

[10] Zeng Y, Xu X, Shen D, et al. (2017) Traffic sign recognition using kernel extreme learning machines with deep perceptual features. IEEE Transactions on Intelligent Transportation Systems, 18(6): 1647-1653.

[11] Liang N Y, Huang G B, Saratchandran P, et al. (2006) A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans on Neural Networks, 17(6): 1411-1423.

[12] ZHANG M Y, WEN Y Y, YANG X T, ZHAO H. (2017) An incremental weighted average based online sequential extreme learning machine algorithm. Control and Decision, 32(10): 1887-1893.

[13] Mao W, Wang J, He L, et al. (2017) Online sequential prediction of imbalance data with two-stage hybrid strategy by extreme learning machine. Neurocomputing, 261(SI): 94-105.

[14] DING S, MIRZA B, LIN Z, et al. (2017) Kernel based online learning for imbalance multiclass classification. Neurocomputing, 277: 139-148.

[15] Shukla S & Raghuvanshi B S. (2019) Online sequential class-specific extreme learning machine for binary imbalanced learning. Neural Networks, 119: 235-248.

[16] Mirza B, Lin Z, Toh K A. (2013) Weighted Online Sequential Extreme Learning Machine for Class Imbalance Learning. Neural processing letters, 38(3): 465-486.

[17] Mathew J, Pang C K, Luo M, et al. (2018) Classification of imbalanced data by oversampling in kernel space of support vector machines. IEEE Transactions on Neural Networks Learning Systems, 1, 1-12.

[18] Wan J W, Yang M. (2020) Survey on cost-sensitive learning method. Ruan Jian Xue Bao/ Journal of Software, 31(1): 113-136.

[19] Zong W, Huang G B, Chen Y Q. (2013) Weighted extreme learning machine for imbalance learning. Neurocomputing, 101: 229-242.

[20] Toh K A, Eng HL. (2008) Between classification error approximation and weighted least-squares learning. IEEE Trans Pattern Anal Mach Intell, 30(4): 658-669.

[21] Toh KA. (2008) Deterministic neural classification. Neural Comput, 20(6): 1565-1595.