Neural Process Lett
DOI 10.1007/s11063-013-9286-9

Weighted Online Sequential Extreme Learning Machine for Class Imbalance Learning

Bilal Mirza · Zhiping Lin · Kar-Ann Toh

Abstract Most of the existing sequential learning methods for class imbalance learn data in chunks. In this paper, we propose a weighted online sequential extreme learning machine (WOS-ELM) algorithm for class imbalance learning (CIL). WOS-ELM is a general online learning method that alleviates the class imbalance problem in both chunk-by-chunk and one-by-one learning. One of the new features of WOS-ELM is that an appropriate weight setting for CIL is selected in a computationally efficient manner. In one-by-one learning of WOS-ELM, a new sample can update the classification model without waiting for a chunk to be completed. Extensive empirical evaluations on 15 imbalanced datasets show that WOS-ELM obtains comparable or better classification performance than competing methods. The computational time of WOS-ELM is also found to be lower than that of the competing CIL methods.

Keywords Class imbalance · Online sequential learning · Extreme learning machine (ELM) · Weighted least squares · Total error rate

1 Introduction

1.1 Background

Class imbalance occurs when the number of samples in one class is much larger than that in the other. A traditional classifier such as extreme learning machine (ELM) [1] which is

B. Mirza · Z. Lin (✉)
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore
e-mail: ezplin@ntu.edu.sg

B. Mirza
e-mail: bilal2@e.ntu.edu.sg

K.-A. Toh
School of Electrical and Electronic Engineering, Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul 120-749, South Korea
e-mail: katoh@yonsei.ac.kr

Published online: 27 February 2013

trained with an imbalance dataset can be biased towards the majority class i.e. it may achieve a very high majority class accuracy by compromising minority class accuracy. Class imbalance learning (CIL) is a challenging problem and has received a lot of attention over the past decade or so for batch learning [2]. Generally speaking, two main approaches exist on tackling the CIL problem, the data sampling approach and the algorithm modification approach. In the data sampling approach, some majority class samples are removed (undersampling) or some artificial minority class samples are added (oversampling) in order to balance the size of the two classes. In undersampling, samples can be removed randomly or using some specific criteria e.g. removing noisy, borderline and redundant samples before removing the more informative samples [3]. In oversampling, minority samples can be randomly duplicated or new samples can be created in the neighborhood of the original minority class samples [4]. There is usually some extra learning cost of data analysis and processing associated with the data sampling methods. In the algorithm modification approach, cost-sensitive learning methods are frequently used to tackle the CIL problem. Cost-sensitive learning methods shift the undesired bias of the majority class towards the more important minority class e.g. by assigning higher misclassification cost or higher weight to the minority class samples [5]. In this paper, minority class is considered as the positive class and majority class as the negative class.

In various machine learning applications, datasets are not completely available at a given time instance as they may arrive sequentially. For such cases, batch learning algorithms are not suitable since both the old and the new data i.e. the complete dataset is used for retraining, every time a new chunk of data arrives. In contrast, sequential or incremental learning algorithms store the previously learned information and update themselves only with the new data. Once the chunk of data has been used for training, it may be discarded. Sequential learning algorithms are preferable over batch learning algorithms not only for their reduced computational time on large datasets but also for their ability of adapting to online learning applications [6,7]. Sequential learning algorithms have shown promising results with balanced classes [6–12]. However, many real-world datasets or data streams are imbalanced, such as in medical diagnosis and fraud detection applications. In biometric scores fusion applications, the number of imposters is far greater than the number of genuine users [13]. Hence, there is a need for online sequential learning methods which can tackle the class imbalance problem.

It is not until recently that some incremental learning algorithms have been developed to address the class imbalance problem [14–20]. Most of these methods integrate data sampling into the incremental learning algorithms. Learn++.SMOTE [14] uses SMOTE [4], one of the most popular oversampling techniques, in the Lean++.NSE [12] incremental learning algorithm. Learn++.UDNC [15] focuses on the CIL problem and has the ability to learn new-classes as well. Gao [16] proposed to collect the minority class samples from all the previous chunks and if required, remove some of the majority class samples from the current chunk in order to balance the two classes. SERA [17] and REA [18] incorporated some of the previous minority class samples into the current data chunk. The previous minority class samples are selected based on the degree of similarity with the current minority class samples. Wang [19] used undersampling by k-Means clustering of the majority class samples. In the context of incremental learning, oversampling is usually preferred over undersampling because of the information loss associated with undersampling [14].

Very recently, an online total error rate (OTER) algorithm is proposed in [13] for learning one sample at a time, or the so-called one-by-one learning. OTER is an approximated online version of the total error rate (TER) method [21,22], which is based on classification error

and is for batch learning. For TER minimization, OTER assigns fixed weights to the samples corresponding to their class sizes. Although the OTER algorithm is developed mainly for biometric scores fusion, it can be used for the class imbalance problem and is closely related to the work presented in this paper.

1.2 Motivation

Since the existing data sampling based sequential learning methods in general learn data in chunks, or the so-called chunk-by-chuck learning, they treat the CIL problem in a way similar to the batch learning methods. A limitation of data sampling methods is that they do not consider one-by-one learning and cannot be directly applied to one-by-one learning easily without accessing the old data. Thus, update of the classification model may be delayed until a complete data chunk is received. This may not be desired in some applications. For example, in classification of emails as relevant or irrelevant to a user, the classification model should be immediately updated as soon as the user identifies/tags an important email [20]. Another problem with data sampling methods is that learning cannot be carried out efficiently with highly imbalanced data streams, when the chunk size is smaller than the imbalance ratio. This is because in such a case, a minority sample may arrive after several chunks comprising of only majority class samples. For most data sampling methods to work, each chunk has to contain some minority samples. Some of the data sampling methods [16–18] can overcome the above problem by relying on previously learnt data which however, may not be always accessible, particularly when the data size is huge [14]. Moreover, the training time for data sampling methods may be long as some extra learning cost is required for data analysis and processing. This may be another limitation as computational efficiency is crucial for online learning.

On the other hand, the OTER method is a computationally efficient online learning algorithm. Similar to online sequential extreme learning machine (OS-ELM) [7], tuning of hidden node parameters is not required in OTER. In order to minimize TER, OTER uses fixed weights which correspond to the class sizes. Specifically, OTER assigns higher weight to the minority class samples than weight to the majority class samples. As a result, OTER generally performs better than traditional online classifiers like OS-ELM when the class sizes are different. The limitation of the OTER algorithm is that it is for one-by-one learning only and it uses fixed weights.

Motivated by the online sequential learning methods discussed above, we aim at developing a general online sequential learning method for the CIL problem. Following the OTER method, we start from the online sequential ELM algorithm by assigning higher weight to the minority class samples than to the majority class samples through a properly designed weighting matrix. However, instead of adopting fixed weights, an efficient weight selection strategy is proposed to obtain better classification performance and at the same time to maintain the computational efficiency. Moreover, the proposed method can learn data either chunk-by-chunk or one-by-one, without accessing the previously learnt data.

1.3 Contributions

We propose a general online sequential learning method for the CIL problem. This method is referred to as weighted online sequential extreme learning machine (WOS-ELM). The main contributions of this paper are enumerated as follows:

1. The proposed WOS-ELM algorithm can tackle the CIL problem in both chunk-by-chunk and one-by-one learning. To the best of our knowledge, this is the first online sequential learning algorithm for the CIL problem with the capability of both chunk-by-chunk and one-by-one learning. Since the proposed method works for any chunk size, a chunk may or may not contain minority class samples.
2. A computationally efficient weight selection procedure for alleviating the class imbalance problem is proposed. The weight tuning is performed to optimize a more appropriate measure for the CIL problem called G_{mean}.
3. Extensive empirical comparison is performed between WOS-ELM and data sampling methods for chunk-by-chunk learning and between WOS-ELM and the OTER method for one-by-one learning. Statistical significance test is also carried out to check whether the new method is statistically better than the methods in comparison in terms of G_{mean}.

This paper is organized as follows. Section 2 discusses the preliminaries. Section 3 presents the details of the new online sequential learning method for class imbalance. This is followed by experiments for validating the performance of the proposed framework in Section 4. Finally, Section 5 concludes the paper.

2 Preliminaries

In this section, we first briefly review the theory of batch ELM [1] and online sequential ELM [7]. Then we discuss the TER method, a classification error based objective function [21,22], and its online version referred to as the OTER method [13].

2.1 ELM and Online Sequential ELM Methods

Extreme learning machine was proposed in [1] which essentially provides a single step least squares error (LSE) estimate solution for the single-hidden-layer feedforward network (SLFN). With ELM there is no need to tune the hidden layer of the SLFN as in traditional gradient based algorithms. Independent of the training data, ELM randomly assigns computational nodes in the hidden layer while the output weights connecting the hidden layer and the output layer are determined analytically.

Consider a q class training dataset $\{x_i, y_i\}$, $i = 1, \ldots, N$ and $y_i \in R^q$. $x_i \in R^d$ is a d-dimensional data point. The SLFN output with L hidden nodes is given by

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

where a_j and b_j, $j = 1, \ldots, L$ are the jth hidden node's learning parameters assigned randomly without considering the input data. $\beta_j \in R^q$ is the output weight vector connecting the jth hidden node to the output nodes and $G(x)$ can be any infinitely differentiable activation function such as sigmoidal function or radial basis function in the hidden layer.

The N equations in (1) can be written in a compact form as below

$$O = H\beta \tag{2}$$

where

$$H = \begin{bmatrix} G(a_1, b_1, x_1) & \ldots & G(a_L, b_L, x_1) \\ \vdots & \ldots & \vdots \\ G(a_1, b_1, x_N) & \ldots & G(a_L, b_L, x_N) \end{bmatrix}_{N \times L}$$

is called the hidden layer output matrix, H_{ij} represents the jth hidden node output corresponding to the input x_i. $\beta = [\beta_1, \beta_2, \ldots, \beta_L]^T$ and $O = [o_1, o_2, \ldots, o_N]^T$.

In order to find the output weight matrix β which minimizes the cost function $\|\|O - Y\|\|$, a LSE solution of (2) is obtained as below

$$\beta = H^{\dagger} Y = (H^T H)^{-1} H^T Y \tag{3}$$

where $Y = [y_1, y_2, \ldots, y_N]^T$ and $H^{\dagger}$ is the Moore-Penrose generalized inverse of matrix H. This closed form single step LSE solution is referred to as extreme learning machine [1].

An online sequential version of batch ELM referred to as online sequential extreme learning machine (OS-ELM) [7] has been proposed for the SLFN. OS-ELM achieves better generalization performance than the previous algorithms proposed for SLFN [8,9] at a much faster learning speed. It can learn data one-by-one as well as chunk-by-chunk (with a fixed or varying chunk length). Only the newly arrived samples are learnt at any given time so the samples that have already been used in the learning procedure can be discarded. The online sequential learning in OS-ELM consists of two steps.

Step1:Initialization

In this step, a small portion of training data $n_0 = \{x_i, y_i\}, i = 1, \ldots, N_0$ with $N_0 \in N$ is considered for initializing the learning. The initial output weight matrix is calculated according to the ELM algorithm by randomly assigning the input weights a_j and bias $b_j, \quad j = 1, \ldots, L$ as follows

$$\beta^{(0)} = P_0 H_0^T Y_0 \tag{4}$$

where $P_0 = (H_0^T H_0)^{-1}$ and H_0 is the initial hidden layer output matrix.

It is recommended that the number of initial training samples should be greater than or equal to the number of hidden neurons. With this setting the generalization performance of online sequential ELM reaches that of the batch ELM.

Step 2: Sequential Learning

Upon the arrival of new set of observations $n_{k+1} = \{x_i, y_i\}$, $i = (\sum_{l=0}^{k} N_l) + 1, \ldots, \sum_{l=0}^{k+1} N_l$, i.e. the $(k+1)$th chunk of data, first compute the partial hidden layer output matrix H_{k+1}. N_{k+1} is the number of samples in the $(k+1)$th chunk. Then by using the output weight update equation shown below, calculate the output weight matrix $\beta^{(k+1)}$ with $Y_{k+1} = [y_{(\sum_{l=0}^{k} N_l)+1}, \ldots, y_{\sum_{l=0}^{k+1} N_l}]^T$.

$$\begin{aligned} P_{k+1} &= P_k - P_k H_{k+1}^T (I + H_{k+1} P_k H_{k+1}^T)^{-1} H_{k+1} P_k \\ \beta^{(k+1)} &= \beta^{(k)} + P_{k+1} H_{k+1}^T (T_{k+1} - H_{k+1} \beta^{(k)}) \end{aligned} \tag{5}$$

The above equation is analogous to the recursive least squares algorithm. Each time a new chunk of data arrives, the output weight matrix is updated according to (5).

Note that the general update Eq. (5) is for both chunk-by-chunk and one-by-one learning since the one-by-one learning can be considered as a special case of chunk-by-chunk learning with chunk size equal to 1.

2.2 TER and OTER Methods

In machine learning applications, least square error has been used as an objective function due to its simplicity and tractability in optimization. LSE has shown good generalization performance, particularly for regression applications [1]. For classification applications, there exists a mismatch between the minimization of cost function in LSE and the actual classification objective. Recently, a classification error based objective function is proposed [21,22]

Table 1 Confusion matrix

	Predicted positive	Predicted negative
Actual positive	tp	fn
Actual negative	fp	tn

which, under quadratic approximation, is deterministically solvable. The proposed objective function essentially minimizes the TER or maximizes the overall accuracy. Accuracy (Acc), a commonly used evaluation criterion of a standard classification system, is defined as

$$\text{Acc} = \frac{tp + tn}{tp + fn + tn + fp} \tag{6}$$

where, tp is the correctly classified positive samples, tn the correctly classified negative samples, fn the misclassified positive samples, fp the misclassified negative samples. Also refer to the confusion matrix defined in Table 1 for the above quantities

Let

$tp + fn = m^+ =$ number of positive class samples
$tn + fp = m^- =$ number of negative class samples
$TPR = tp/m^+ =$ true positive rate
$TNR = tn/m^- =$ true negative rate
$FPR = fp/m^- =$false positive rate
$FNR = fn/m^+ =$ false negative rate
$s = m^-/m^+$

Now (6) becomes

$$\text{Acc} = \frac{TPR + sTNR}{1 + s} \tag{7}$$

In order to relate Acc with TER, s is set to 1 (the relative importance of TPR and TNR is the same) and also the normalization factor is ignored [22]. Thus, (7) becomes

$$\begin{aligned} \text{Acc} &= TPR + TNR \\ &= \frac{tp}{m^+} + \frac{tn}{m^-} \\ &= 1 - TER \end{aligned} \tag{8}$$

where

$$TER = FPR + FNR \tag{9}$$

Under quadratic approximation, solution for direct minimization of TER was found to be related to that of weighted least squares method with class-specific weight setting corresponding to the sizes of the two classes (m^-, m^+). The output weight matrix β that minimizes the TER is found to be [22]

$$\beta = (H^T W H)^{-1} H^T W Y \tag{10}$$

where $W = W_- + W_+$ and $W_- = diag(\frac{1}{m^-}, \ldots, \frac{1}{m^-}, 0, \ldots, 0) W_+ = diag(0, \ldots, 0, \frac{1}{m^+}, \ldots, \frac{1}{m^+})$.

Note that the elements of H and Y are ordered according to the samples in the two classes i.e. all the negative class samples come first, followed by the positive class samples.

It can be observed that (10) has a similar structure as (3), except for the existence of a weighting matrix W. In this paper, we call this weighted least squares solution as weighted extreme learning machine (WELM). By setting $m^+ = m^- = 1$, W becomes an identity matrix and Eqs. (10) reduces to (3).

The main reason for the superior performance of the TER method over least square error classifier is its ability to cater for difference in the positive and negative class sizes through the weighting matrix.

Recently, an approximated online version of TER i.e. OTER is proposed for one-by-one learning. Similar to online sequential ELM, learning in OTER consists of two steps; initialization and sequential learning. The output weight update equation is given as

$$\begin{aligned} P_{k+1} &= (\mu_{k+1})^{-1} P_k - \frac{(\mu_{k+1})^{-1} P_k h_{k+1} h_{k+1}^T (\mu_{k+1})^{-1} P_k}{w_{k+1}^{-1} + h_{k+1}^T (\mu_{k+1})^{-1} P_k h_{k+1}} \\ \beta^{(k+1)} &= \beta^{(k)} + w_{k+1} P_{k+1} h_{k+1} (t_{k+1} - h_{k+1}^T \beta^{(k)}) \end{aligned} \tag{11}$$

where

$$w_{k+1} = \begin{cases} w_{k+1}^- = \dfrac{1}{m_{k+1}^-} & \text{if } t_{k+1} = -1 \\ w_{k+1}^+ = \dfrac{1}{m_{k+1}^+} & \text{if } t_{k+1} = +1 \end{cases}$$

with

m_{k+1}^- = no. of negative class samples received up till $(k+1)$th iteration
m_{k+1}^+ = no. of positive class samples received up till $(k+1)$th iteration

and stacking weight update

$$\mu_{k+1} = \begin{cases} \dfrac{1}{2}(1 + \dfrac{w_{k+1}^-}{w_k^-}) & \text{if } t_{k+1} = -1 \\ \dfrac{1}{2}(1 + \dfrac{w_{k+1}^+}{w_k^+}) & \text{if } t_{k+1} = +1 \end{cases}$$

In order to minimize TER, the weights associated with both the old and the new samples should correspond to the current class sizes. μ_{k+1}provides an approximate rescaling of the weights associated with the old samples.

3 Proposed Method

In this section, we propose a WOS-ELM algorithm for class imbalance learning. WOS-ELM can learn data in both chunk-by-chunk and one-by-one mode. In WOS-ELM, the weights for alleviating the class imbalance problem are tuned in a computationally efficient manner. Before formally proposing WOS-ELM, we briefly discuss the effect of weights tuning in weighted extreme learning machine (WELM).

3.1 Weight Selection in Weighted ELM for Class Imbalance Learning

Weighted ELM can be considered as a cost-sensitive algorithm for CIL since it assigns higher weight to the minority class samples i.e. $\frac{1}{m^+} > \frac{1}{m^-}$. This particular setting ($w^- = \frac{1}{m^-}$ and $w^+ = \frac{1}{m^+}$) in (8) assigns equal weight to the positive class and the negative class accuracies (TPR and TNR). One can tune the weights of WELM to optimize a suitable measure for CIL.

It should be first pointed out that the overall accuracy or TER may not be an appropriate evaluation criterion for highly imbalanced classes. Traditional classifiers like ELM can achieve very high overall accuracy by obtaining very high TNR at the expense of very low TPR. This is not desirable in applications such as medical diagnosis of a rare disease. In this application, correct classification of the minority class samples is usually more important than that of the majority class samples. For example, a dataset may contain only 5 % of minority class samples and 95 % of majority class samples. If the classifier classifies all the samples as belonging to the majority class, the overall accuracy of 95 % can be obtained but the accuracy for the minority class will be 0 %. In such a case, although a very low TER (or very high overall accuracy) is obtained, the important minority class is ignored completely. To overcome this problem, G_{mean} is adopted as a measure for the classification performance in this paper and it is defined below:

$$G_{\text{mean}} = \sqrt{TPR \times TNR}$$

In fact, G_{mean} is a popular measure for the CIL problem [2,23–25]. It is used when the combined performance of the two classes needs to be judged by a single metric. It measures both TPR and TNR in a balanced way. Unlike the overall accuracy, a high G_{mean} value requires both TPR and TNR to be high and the difference between TPR and TNR to be low. For highly imbalanced datasets, a high overall accuracy can always be achieved by classifying all the samples into the majority class and ignoring the minority class samples completely. On the other hand, G_{mean} cannot be high if the accuracy of one of the two classes is very low. Consider the above simple example again, G_{mean} will become zero when all the samples are classified as in the majority class.

Depending on the applications, other evaluation measures such as area under ROC curve (*auROC)*, F-measure, and Adjusted G_{mean} (AG_{mean}) etc. can be optimized instead of G_{mean} e.g. in some bioinformatics applications where a high sensitivity of change detection in TNR as compared to that in TPR is required, the Adjusted G_{mean} or AG_{mean} can be more meaningful than G_{mean} [26–28]. However, our work is closely related to the total error rate method which may have the short-coming of ignoring TPR in order to maximize the overall accuracy. Thus in this work, we aim at increasing TPR and TNR in a balanced way, i.e., considering both being equally important. This can be achieved by G_{mean} optimization with efficient weight tuning which will be discussed shortly. In our future work, we shall consider formulating efficient weight tuning for the optimization of other CIL evaluation measures as well.

For the minimization of TER, weight tuning is proposed [22] using Table 2. In most of the two class datasets used in [21,22], the ratio of the two class sizes (imbalance ratio) is between one and two and the highest imbalance ratio is 3.2 i.e. the datasets are either balanced or slightly imbalanced.

We have applied the weight setting suggested in Table 2 above [22] and found that the best weight setting from the table may not lead to the highest G_{mean}. The weights setting $M = 1$ assigns equal weight to TPR and TNR i.e. it maximizes

Table 2 Weight setting for TER [22]

M	$(\frac{1}{w^-}, \frac{1}{w^+})$	M	$(\frac{1}{w^-}, \frac{1}{w^+})$
0	$(1, 1)$	5	$(0.5m^-, m^+)$
1	(m^-, m^+)	6	$(m^-, 2m^+)$
2	$(m^-, 1)$	7	$(2m^-, m^+)$
3	$(1, m^+)$	8	$(m^-, 0.2m^+)$
4	$(m^-, 0.5m^+)$	9	$(0.2m^-, m^+)$

both accuracies with equal importance. Thus, for obtaining a higher G_{mean} for a class imbalanced dataset, we may need to tune near this particular setting. From [21,22], we selected the dataset with the highest imbalance ratio (3.2) and three datasets with imbalance ratio between 1 and 2. G_{mean} is maximized using the weight settings in Table 2. A finer tuning is then performed with an interval of 0.1 between $M = 1$ and its closet settings $M = 6$ and $M = 7$. The weight settings used for fine tuning are (m^-, m^+), $(1.1m^-, m^+)$, $(1.2m^-, m^+)$, $\ldots$, $(2m^-, m^+)$, $(m^-, 1.1m^+)$, $(m^-, 1.2m^+)$, $\ldots$, $(m^-, 2m^+)$. 50 runs of 10 fold cross-validation is performed with the average number of hidden neurons reported in [22]. On 3 out of 4 datasets, the highest G_{mean} is obtained using fine tuning.

We also evaluated the performance of weighted ELM on 15 real-world imbalanced datasets. These datasets will later be used for online sequential learning. On eight out of fifteen datasets, higher G_{mean} values than those obtained by $M = 1$ are observed using fine tuning. On four datasets, the improvement is higher than 1 % while on the remaining four datasets the improvement is less than 1 %. Though the improvement is small, it cannot be ignored. On these 15 imbalanced datasets, weight settings $M = 6$ and $M = 7$ do not obtain higher G_{mean} than $M = 1$. Note that $M = 0$ represents original ELM which obtained lower G_{mean} than $M = 1$ for all these highly imbalanced datasets. From the results, we recognize the importance of tuning the weights of WELM in the vicinity of $M = 1$.

Note that an exhaustive weight tuning is performed in [22] for minimization of TER i.e. all the ten weights settings are always tested. In contrast, in our proposed method, when G_{mean} is maximized, the result of $M = 1$ are considered first in order to test the next appropriate setting. This efficient weight tuning is adopted for online sequential learning in the next sub-section.

3.2 Weighted Online Sequential Extreme Learning Machine

After recognizing the importance of tuning weights in Weighted ELM, we now discuss an efficient sub-optimal weight selection method for alleviating the class imbalance problem in online learning. The recent online total error rate method [13] uses fixed weights aiming at the minimization of total error rate. In the WOS-ELM to be described shortly, weight tuning is adopted to optimize the G_{mean} which is an appropriate CIL evaluation criterion as discussed previously. To compromise classification performance and computational efficiency, the fine-tuning suggested in the previous sub-section is further simplified.

We start with $M = 1$ (see Table 2) as it maximizes TPR and TNR with equal weight, i.e. the accuracies of both classes are maximized without a bias. By observing the TPR and TNR obtained, further weight setting is considered in an attempt to achieve a higher G_{mean}. In general, G_{mean} is higher when both accuracies are equal. Hence, if TPR and TNR are already close using $M = 1$, there is no need to tune the weights further. Now if TPR is greater than TNR, we will try to increase TNR by tuning the weights in the

range between (m^-, m^+) and $(m^-, 2m^+)$. For sub-optimal tuning to keep the computational cost low, we only select $(m^-, 1.2m^+)$, $(m^-, 1.5m^+)$, $(m^-, 1.8m^+)$, $(m^-, 2m^+)$ to increase TNR. Similarly, if TNR is greater than TPR, we only select $(1.2m^-, m^+)$, $(1.5m^-, m^+)$, $(1.8m^-, m^+)$, $(2m^-, m^+)$ to increase TPR. For most of the imbalanced datasets we have tested, the above suggested weight tuning is sufficient.

We now describe the WOS-ELM in details. Following (10), the weighted least square solution corresponding to the initialization chunk n_0 is given by

$$\beta^{(0)} = K_0^{-1} H_0^T W_0 Y_0 \tag{12}$$

$$K_0 = H_0^T W_0 H_0 \tag{13}$$

where W_0 is constructed according to the size of the two classes in n_0. The starting weights are given as $w_0^- = \frac{1}{m_0^-}$ and $w_0^+ = \frac{1}{m_0^+}$, where m_0^- and m_0^+ represent number of negative class samples and number of positive class samples in the chunk respectively. w_0^- is the weight associated with all the majority class samples and w_0^+ is the weight associated with all the minority class samples in this chunk. Alternatively, to simplify the weight tuning, w_0^- can be set as 1 and w_0^+ can be set as the current imbalance ratio i.e. $\frac{m_0^-}{m_0^+}$. By observing TPR and TNR obtained using the validation set, w_0^+ is increased or decreased for optimization as discussed above, e.g. to increase TPR, $1.2w_0^+$, $1.5w_0^+$, $1.8w_0^+$, $2w_0^+$ can be tuned as an attempt to increase G_{mean}.

With the arrival of the next chunk of data, the weighted least square solution is given by

$$\beta^{(1)} = K_1^{-1} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} W_0 & 0 \\ 0 & W_1 \end{bmatrix} \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} \tag{14}$$

$$K_1 = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} W_0 & 0 \\ 0 & W_1 \end{bmatrix} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}$$

where H_1, Y_1 correspond to the second chunk of data n_1. The starting weights in W_1 are given as $w_1^- = 1$ and $w_1^+ = \frac{m_0^- + m_1^-}{m_0^+ + m_1^+}$, where w_1^+ represents the overall imbalance ratio and m_1^- and m_1^+ represent the numbers of negative class samples and positive class samples in the new chunk, respectively. As in the first chunk, w_1^+ is also tuned for a possible higher G_{mean}

The purpose of W_0 and W_1 is to assign higher weight to the minority class samples. Let us assume that the imbalance ratio is the same in both chunks, i.e. $w_0^+ = w_1^+ = w^+$. It is possible that the weight associated with the minority class samples in the first chunk that maximizes G_{mean} is $1.2w^+$ while in the second chunk it is $1.5w^+$. For G_{mean} optimization, we can either allow different minority class weights in different chunks or use the same weight in all the chunks. For using the same weight, a rescaling of the old weights has to be performed. For an exact rescaling, an inversion of the summation of three matrices is required which is mathematically difficult, if not impossible to achieve (see [13]). In OTER, an approximate rescaling is proposed [13]. It reduces three terms to two terms such that the matrix inversion lemma can be utilized. In OTER, rescaling is required since a common weight setting corresponding to the class sizes needs to be associated with all the samples. In WOS-ELM, we use different weights in different chunks to maximize G_{mean}. Thus, instead of performing approximate weights rescaling of the previously learnt samples, we allow

different minority class weights in different chunks. With this in mind, we are ready to derive the online sequential algorithm in the following.

For sequential learning $\beta^{(1)}$ should be expressed as a function of K_1, H_1, Y_1 and not as a function of old dataset K_0, H_0, Y_0.

$$K_1 = [\, H_0^T \;\; H_1^T \,] \begin{bmatrix} W_0 & 0 \\ 0 & W_1 \end{bmatrix} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} = H_0^T W_0 H_0 + H_1^T W_1 H_1 = K_0 + H_1^T W_1 H_1$$

Similarly,

$$\begin{aligned} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} W_0 & 0 \\ 0 & W_1 \end{bmatrix} \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} &= H_0^T W_0 Y_0 + H_1^T W_1 Y_1 \\ &= K_0 K_0^{-1} H_0^T W_0 Y_0 + H_1^T W_1 Y_1 \\ &= K_0 \beta^{(0)} + H_1^T W_1 Y_1 \\ &= (K_1 - H_1^T W_1 H_1)\beta^{(0)} + H_1^T W_1 Y_1 \\ &= K_1 \beta^{(0)} - H_1^T W_1 H_1 \beta^{(0)} + H_1^T W_1 Y_1 \end{aligned}$$

Now,

$$\begin{aligned} \beta^{(1)} &= K_1^{-1} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} W_0 & 0 \\ 0 & W_1 \end{bmatrix} \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} \\ &= K_1^{-1}(K_1 \beta^{(0)} - H_1^T W_1 H_1 \beta^{(0)} + H_1^T W_1 Y_1) \\ &= \beta^{(0)} + K_1^{-1} H_1^T W_1 (Y_1 - H_1 \beta^{(0)}) \end{aligned}$$

The above arguments can be generalized as follows

$$\begin{aligned} K_{k+1} &= K_k + H_{k+1}^T W_{k+1} H_{k+1} \\ \beta^{(k+1)} &= \beta^{(k)} + K_{k+1}^{-1} H_{k+1}^T W_{k+1} (Y_{k+1} - H_{k+1} \beta^{(k)}) \end{aligned}$$

$$W_{k+1} = diag(\begin{bmatrix} w_{k+1} \dots w_{k+1} \end{bmatrix}) \in R^{\left(\sum_{l=0}^{k+1} N_l\right) \times \left(\sum_{l=0}^{k+1} N_l\right)} \text{ with } w_{k+1} \in \{w_{k+1}^-, w_{k+1}^+\}$$

We used K_{k+1}^{-1} instead of K_{k+1} for the computation of $\beta^{(k+1)}$. The modified update expression for K_{k+1}^{-1} using matrix inversion lemma is given by

$$\begin{aligned} K_{k+1}^{-1} &= (K_k + H_{k+1}^T W_{k+1} H_{k+1})^{-1} \\ &= K_k^{-1} - K_k^{-1} H_{k+1}^T (W_{k+1}^{-1} + H_{k+1} K_k^{-1} H_{k+1}^T)^{-1} H_{k+1} K_k^{-1} \end{aligned}$$

Let $P_{k+1} = K_{k+1}^{-1}$, the update equation for WOS-ELM become

$$\begin{aligned} P_{k+1} &= P_k - P_k H_{k+1}^T (W_{k+1}^{-1} + H_{k+1} P_k H_{k+1}^T)^{-1} H_{k+1} P_k \\ \beta^{(k+1)} &= \beta^{(k)} + P_{k+1} H_{k+1}^T W_{k+1} (Y_{k+1} - H_{k+1} \beta^{(k)}) \end{aligned} \tag{15}$$

Every time a new sample or a new chunk of samples arrives, W_{k+1} is selected to maximize G_{mean}.

Since w_{k+1}^- is always set to 1, no tuning is required upon the arrival of majority class samples in one-by-one learning. Similarly, no tuning is required if the arriving chunk does not

contain any minority class sample. The WOS-ELM method is summarized in Algorithm 1. As in the online sequential ELM algorithm, the general update Eq. (15) works for both chunk-by-chunk and one-by-one learning since the one-by-one learning can be considered as a special case of chunk-by-chunk learning with chunk size equal to 1.

Algorithm 1: Weighted online sequential extreme learning machine (WOS-ELM) algorithm (training)

Input: $\{x_i, y_i\}, x_i \in R^d$ and $y_i \in R^q, i = 1, \ldots\ldots, N$.
Output: $\beta^{(k+1)}$.

Initialization:
Initial data set: $n_0 = \{x_i, y_i\}, x_i \in R^d$ and $y_i \in R^q, i = 1, \ldots\ldots, N_0$ with $N_0 \in N$.
1. Set hidden nodes L using cross-validation ($L \leq N_0$).
2. Random hidden node parameters: $a_j \in R^d$ and $b_j \in R^1$ $j = 1, .., L$.
3. Initial hidden layer output matrix $H_0 = \begin{bmatrix} G(a_1, b_1, x_1) & \ldots & G(a_L, b_L, x_1) \\ \vdots & \ldots & \vdots \\ G(a_1, b_1, x_{N_0}) & \ldots & G(a_L, b_L, x_{N_0}) \end{bmatrix}_{N_0 \times L}$
4. Initial diagonal weighting matrix W_0 with $w_0^- = 1$ and $w_0^+ = \frac{m_0^-}{m_0^+}$ (i.e. imbalance ratio of n_0)
$$W_0 = diag([w_0 \ \ldots \ w_0]) \in \mathrm{R}^{N_0 \times N_0} \text{ with } w_0 \in \{w_0^-, w_0^+\}$$
5. Inverse of initial $K_0 \rightarrow (K_0)^{-1} = (H_0^T W_0 H_0)^{-1}$.
6. Estimate initial solution $\beta^{(0)}$ using (12).
7. Find G_{mean}, TPR and TNR using $\beta^{(0)}$ on the validation set.
if $TPR = TNR$ **then** %% We consider $TPR = TNR$ if $|TPR - TNR| \leq 1\%$ %%
Keep $(w_0^-, w_0^+) = (1, \frac{m_0^-}{m_0^+})$.
else if $\{TPR > TNR\}$ **then**
Tune $w_0^+ \rightarrow \frac{w_0^+}{1.2}, \frac{w_0^+}{1.5}, \frac{w_0^+}{1.8}$, and $\frac{w_0^+}{2}$.
Select $\beta^{(0)}$ and $(K_0)^{-1}$ yielding highest G_{mean}.
else
Tune $w_0^+ \rightarrow 1.2w_0^+, 1.5w_0^+, 1.8w_0^+$ and $2w_0^+$.
Select $\beta^{(0)}$ and $(K_0)^{-1}$ yielding highest G_{mean}.
end if

Sequential Learning:
New data chunk: $n_{k+1}=\{x_i, y_i\}, i = (\sum_{l=0}^{k} N_l) + 1, \ldots, \sum_{l=0}^{k+1} N_l$.
1. Calculate the hidden layer output matrix H_{k+1}
2. Calculate the diagonal weighting matrix W_{k+1}:
$$W_{k+1} = diag([w_{k+1} \ \ldots \ w_{k+1}]) \in \mathrm{R}^{\left(\sum_{l=0}^{k+1} N_l\right) \times \left(\sum_{l=0}^{k+1} N_l\right)} \text{ with } w_{k+1} \in \{w_{k+1}^-, w_{k+1}^+\}$$
$$w_{k+1}^- = 1 \ , w_{k+1}^+ = \frac{\sum_{i=0}^{k+1} m_i^-}{\sum_{i=0}^{k+1} m_i^+}$$
3. Find P_{k+1} and New estimate $\beta^{(k+1)}$ using (15)
4. Find G_{mean}, TPR and TNR using $\beta^{(k+1)}$ on the validation set.
if $TPR = TNR$ **then** %% We consider $TPR = TNR$ if $|TPR - TNR| \leq 1\%$ %%
Keep $(w_{k+1}^-, w_{k+1}^+) = (1, \frac{\sum_{i=0}^{k+1} m_i^-}{\sum_{i=0}^{k+1} m_i^+})$
else if $\{TPR > TNR\}$ **then**
Tune $w_{k+1}^+ \rightarrow \frac{w_{k+1}^+}{1.2}, \frac{w_{k+1}^+}{1.5}, \frac{w_{k+1}^+}{1.8}$, and $\frac{w_{k+1}^+}{2}$
Select $\beta^{(k+1)}$ and P_{k+1} yielding highest G_{mean}.
else
Tune $w_{k+1}^+ \rightarrow 1.2w_{k+1}^+, 1.5w_{k+1}^+, 1.8w_{k+1}^+$ and $2w_{k+1}^+$.
Select $\beta^{(k+1)}$ and P_{k+1} yielding highest G_{mean}.
end if

Table 3 Details of the datasets

Dataset	No. of attributes	Imbalance ratio	No. of training samples	No. of testing samples	No. of validation samples
Page-block5	10	46.59	3473	1,000	1,000
Abalone15	8	39.55	2,677	1,000	500
Yeast5	8	28.09	1,034	300	150
Car3	6	24.04	1,228	300	200
Oil	49	21.85	637	200	100
Abalone9-18	8	16.4	531	100	100
Balance2	4	11.75	385	150	90
Satimage4	36	9.27	5,835	600	600
mf-morph10	6	9	1,500	300	200
mf-zernike10	47	9	1,500	300	200
cmc2	9	3.42	973	300	200
Transfusion	4	3.2	548	200	100
Haberman	3	2.77	246	60	40
Phoneme	5	2.4	4,204	600	600
Pima	8	1.86	668	100	100

In summary, the significance of the new method is that it can tackle the CIL problem in chunk-by-chunk as well as one-by-one sequential learning mode. Unlike most data sampling methods, each arriving sample can update the classification model independently without any delay. The previously learnt minority class samples need not be stored in the system. It can also efficiently alleviate the CIL problem when the imbalance ratio is higher than the chunk size i.e. minority class samples may not be present in most of the chunks.

In the newly proposed method, weights are tuned in a computationally efficient manner for alleviating the class imbalance problem. OTER is originally proposed for one-by-one learning and uses fixed weight setting for TER minimization while WOS-ELM covers both chunk-by-chunk and one-by-one learning and tune weights for G_{mean} optimization.

Note that WOS-ELM assumes the class concepts do not change over time. The focus of WOS-ELM in this work is on the class imbalance problem in online learning. In our future work, we shall consider problems associated with concept drift [29] by extending WOS-ELM to non-stationary learning.

4 Experiments

4.1 Datasets

In this section, 15 real-world imbalanced datasets have been used. 14 are selected from the UCI repository [30] and the oil dataset is provided by Dr. Holte [31]. Table 3 shows the details of the datasets. Each dataset has been partitioned in to a training set, a validation set and a testing set with approximately the same imbalance ratio. Using these datasets we have covered various sizes, imbalanced ratios and number of features. Some of the datasets, including haberman, transfusion, phoneme, oil and pima have only two classes. For abalone9-18, we selected class 18 and 9 as the minority and the majority class respectively. For the rest of the datasets, a single class is chosen as the minority class and the rest of them are combined to make up the majority class. These datasets have been frequently used for the performance evaluation of various CIL algorithms in batch mode.

Table 4 Number of hidden neurons and size of initialization set

Dataset	No. of hidden neurons	Initialization set size
Page-block5	36	500
Abalone15	19	500
Yeast5	44	300
car3	190	300
Oil	66	200
Abalone9-18	26	100
Balance2	40	90
Satimage4	130	300
mf-morph10	21	300
mf-zernike10	149	300
cmc2	100	200
Transfusion	14	100
Haberman	9	120
Phoneme	170	300
Pima	40	200

4.2 Experimental Settings

All the experiments are performed in Matlab by modifying ELM [1] and OS-ELM [7] packages. These experiments are carried out in a PC with 2.66 GHz CPU and 4 GB RAM.

For each dataset, the number of hidden neurons is gradually increased and the optimal number is then selected by performing 50 runs of 10-fold cross validation on the training dataset. In all the experiments, sigmoid is used as the activation function. Threshold 'τ' is set to 0 and bias 'η' is set to 1 as in [22]. All the input data is normalized to be within $[-1, 1]$ interval. Due to random settings in the ELM hidden node, all the experiments are repeated 50 times with randomly shuffling the training, the validation and the testing samples. For data sampling methods, the entire procedure is repeated five times i.e. in total 250 runs are performed for oversampling and undersampling methods. We have used SMOTE [4] as the oversampling method and Informative Undersampling [3] as the undersampling method. Informative undersampling removes noisy and borderline samples before removing more informative samples. For SMOTE, the value of k is set to 5. If the chunk contains fewer minority class samples, the value of k is decreased accordingly. Whenever a comparison is performed, the number of hidden neurons, random weights and bias, activation function, initialization set and training, validation and testing samples are kept the same. The size of the initialization set for sequential learning and the number of hidden neurons for each dataset are recorded in Table 4. As recommended, the size of the initialization set in online sequential learning is set greater than the number of hidden neurons [7].

4.3 Classification Results

4.3.1 Chunk-by-Chunk Learning

We first compare the performance of WOS-ELM, SMOTE Oversampling with OS-ELM (OS-ELM- SMOTE) and Informative Undersampling with OS-ELM (OS-ELM-UNDER) on all the 15 datasets in chunk-by-chunk mode. The reason for implementing SMOTE and informative undersampling with OS-ELM for online learning is that these two methods

Table 5 Classification results comparison for chunk-by-chunk learning

Dataset	Chunk size	Algorithm	G_{mean}	Training time (s)
Page-block5	100	WOS-ELM	**91.7**	**13.482**
		OS-ELM-SMOTE	91.5	21.241
		OS-ELM-UNDER	52.7	18.325
	500	WOS-ELM	**91.7**	**8.594**
		OS-ELM-SMOTE	91.6	39.64
		OS-ELM-UNDER	50.5	27.74
Abalone15	100	WOS-ELM	**70.9**	**5.058**
		OS-ELM-SMOTE	69.8	8.081
		OS-ELM-UNDER	70.5	9.846
	500	WOS-ELM	**70.9**	**5.55**
		OS-ELM-SMOTE	70.8	10.488
		OS-ELM-UNDER	61	6.507
Yeast5	75	WOS-ELM	**80.4**	**1.911**
		OS-ELM-SMOTE	79.2	2.486
		OS-ELM-UNDER	52.5	3.27
	150	WOS-ELM	**80.2**	**1.412**
		OS-ELM-SMOTE	78.6	2.277
		OS-ELM-UNDER	52.2	2.655
car3	50	WOS-ELM	**95.1**	**3.878**
		OS-ELM-SMOTE	93.4	8.257
		OS-ELM-UNDER	52	4.759
	100	WOS-ELM	**95**	**2.367**
		OS-ELM-SMOTE	94.1	5.652
		OS-ELM-UNDER	53.4	3.734
Oil	50	WOS-ELM	79.9	**1.519**
		OS-ELM-SMOTE	**80.6**	5.217
		OS-ELM-UNDER	54.7	2.968
	100	WOS-ELM	79.5	**0.949**
		OS-ELM-SMOTE	**80.4**	2.751
		OS-ELM-UNDER	59.6	2.539
Abalone9-18	50	WOS-ELM	**84.2**	**0.912**
		OS-ELM-SMOTE	77.9	3.454
		OS-ELM-UNDER	65.3	1.834
	100	WOS-ELM	**84.3**	**0.854**
		OS-ELM-SMOTE	77.1	1.046
		OS-ELM-UNDER	63.1	1.479
Balance2	30	WOS-ELM	**67.6**	**0.812**
		OS-ELM-SMOTE	63.5	1.687
		OS-ELM-UNDER	48.7	1.515
	45	WOS-ELM	**67.7**	**0.638**
		OS-ELM-SMOTE	63.1	1.462
		OS-ELM-UNDER	48.7	1.391

are commonly used data sampling techniques for tackling the CIL problem. SMOTE has recently shown good performance in online learning also [14]. Also, these methods can be implemented without accessing the previously learnt data. The results obtained by OS-ELM are not shown since it obtains lower G_{mean} than the other three methods for all the datasets. Two different chunk sizes are considered for each dataset. The size is big enough such that it has at least few minority class samples, as required by the data sampling methods. Also, the size is not too big so that we have at least few chunks for sequential learning. For sampling methods, the chunk size represents the number of original samples (before sampling). G_{mean} values and the training times for the two chunk sizes are reported in Tables 5 and 6. The

Table 6 Classification results comparison for chunk-by-chunk learning

Dataset	Chunk size	Algorithm	G_{mean}	Training time (s)
Satimage4	50	WOS-ELM	**85.7**	**24.486**
		OS-ELM-SMOTE	85.6	33.364
		OS-ELM-UNDER	57.2	29.966
	300	WOS-ELM	**86.2**	**11.599**
		OS-ELM-SMOTE	85.7	25.456
		OS-ELM-UNDER	56.4	18.093
mf-morph10	50	WOS-ELM	**92.3**	**2.147**
		OS-ELM-UNDER	92.1	10.064
		OS-ELM-UNDER	86.7	5.158
	200	WOS-ELM	**92.3**	**1.613**
		OS-ELM-SMOTE	92.1	2.894
		OS-ELM-UNDER	81.8	2.798
mf-zernike10	50	WOS-ELM	87.8	**5.664**
		OS-ELM-SMOTE	**88.4**	16.924
		OS-ELM-UNDER	56.1	8.047
	200	WOS-ELM	87.5	**3.366**
		OS-ELM-SMOTE	**88.3**	5.049
		OS-ELM-UNDER	55.3	6.077
cmc2	50	WOS-ELM	**66.1**	**2.756**
		OS-ELM-SMOTE	65.6	3.939
		OS-ELM-UNDER	50.2	4.559
	100	WOS-ELM	**66**	**1.869**
		OS-ELM-SMOTE	65.8	3.434
		OS-ELM-UNDER	50.6	3.991
Transfusion	20	WOS-ELM	**67.9**	**0.306**
		OS-ELM-SMOTE	66.9	1.33
		OS-ELM-UNDER	66.7	0.759
	50	WOS-ELM	**67.9**	**0.554**
		OS-ELM-SMOTE	67.1	0.932
		OS-ELM-UNDER	66.5	1.142
Haberman	20	WOS-ELM	63.7	**0.131**
		OS-ELM-SMOTE	**63.9**	0.607
		OS-ELM-UNDER	60.5	0.355
	60	WOS-ELM	**63.6**	**0.207**
		OS-ELM-SMOTE	63.5	0.476
		OS-ELM-UNDER	60.5	0.338
Phoneme	50	WOS-ELM	**84.1**	**12.007**
		OS-ELM-SMOTE	83.8	19.09
		OS-ELM-UNDER	81.8	16.7
	300	WOS-ELM	**84.2**	**9.432**
		OS-ELM-SMOTE	83.8	12.322
		OS-ELM-UNDER	81.2	10.32
Pima	50	WOS-ELM	75.6	**0.305**
		OS-ELM-SMOTE	**76.5**	0.741
		OS-ELM-UNDER	73.4	0.490
	100	WOS-ELM	76.1	**0.209**
		OS-ELM-SMOTE	**76.4**	0.532
		OS-ELM-UNDER	73.6	0.442

imbalance ratios in all the chunks are similar to that of the overall imbalance ratio. For data sampling methods, various minority class distributions e.g. 35, 40, 45, 50, 55, 60, and 65 are tried for the highest G_{mean} using the validation set in each chunk. Similarly, in WOS-ELM weights are tuned in each chunk for the highest G_{mean} using the same validation set. For a fair

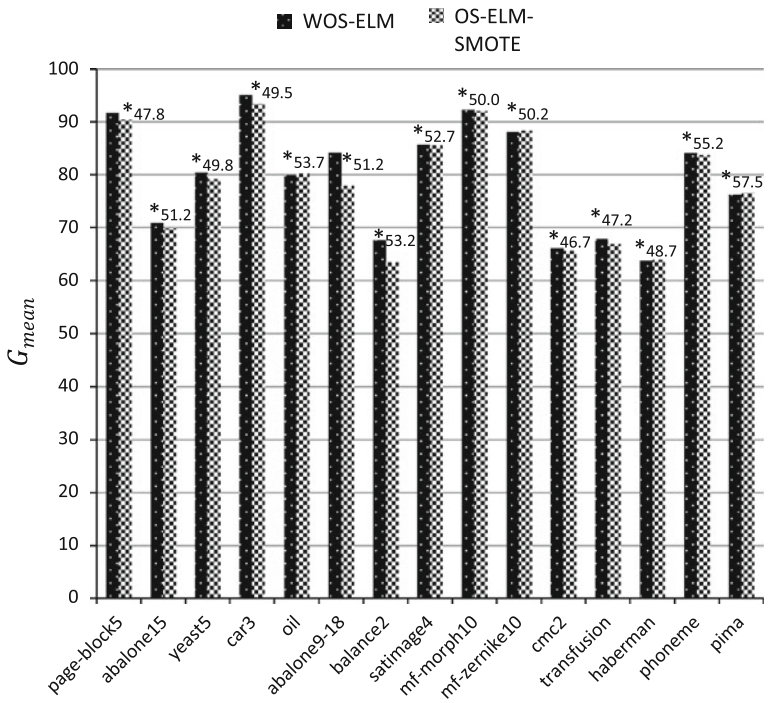

Fig. 1 G_{mean} comparison between WOS-ELM and OS-ELM-SMOTE. Note that * indicates the average minority class distribution in OS-ELM-SMOTE

comparison, undersampling/SMOTE percentages and weights settings are optimized under identical settings. Note that the best G_{mean} values are shown in bold face in tables 5, 6 and 7. The best training time for chunk-by-chunk learning is also shown in bold face in tables 5 and 6. As can be observed from the tables, among the three CIL methods, OS-ELM-UNDER always produce the lowest G_{mean}. While WOS-ELM and OS-ELM-SMOTE produce quite close G_{mean} values with WOS-ELM winning on most datasets. WOS-ELM obtains similar G_{mean} values with different chunk sizes. We also used a chunk size of 10 for three highly imbalanced datasets i.e. page-block5, abalone15 and yeast5 so that a minority class sample arrives after few chunks comprising of only majority class samples. e.g. with page-block5, after every 4 or 5 chunks of all the majority class samples, a chunk arrives with only one minority class sample. Same as in the case of one-by-one learning, the weights are tuned only when the chunk contains at least one minority class sample since majority class samples are assigned a weight of 1. The G_{mean} values obtained in this case are similar to the bigger chunk sizes. WOS-ELM does not require minority class sample(s) to be present in every chunk. A comparison of G_{mean} values obtained by WOS-ELM and OS-ELM-SMOTE is shown in Fig. 1.

WOS-ELM has the lowest training time among the three CIL methods for both chunk sizes. The data sampling methods in comparison use sophisticated sample removal/creation procedures in every chunk which increase the training time. Before the training data is fed to the OS-ELM algorithm, it has to be analyzed and processed e.g. in SMOTE, k-NN rule is used to create synthetic samples. In WOS-ELM, there is no extra cost for data processing.

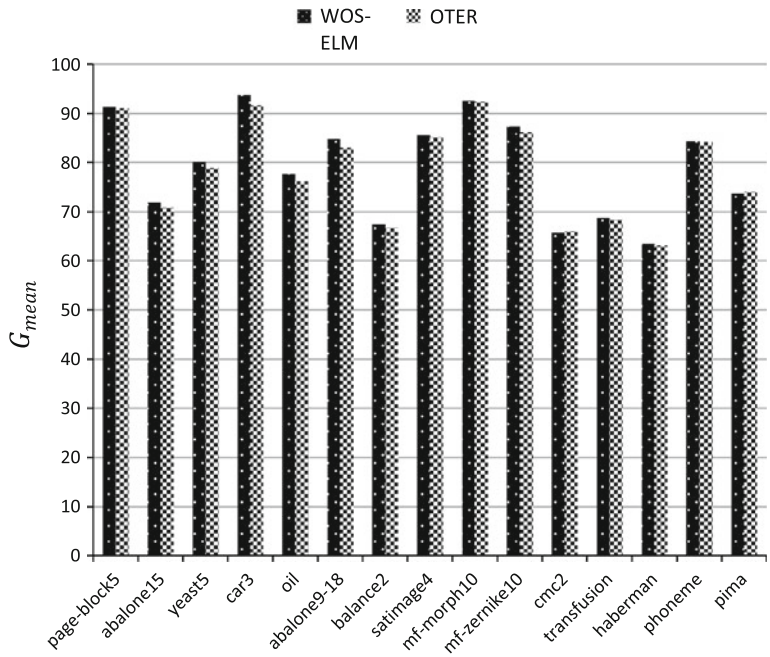

Fig. 2 G_{mean} comparison between WOS-ELM and OTER

4.3.2 One-by-One Learning

Unlike OS-ELM-SMOTE and OS-ELM-UNDER methods, WOS-ELM can learn one sample at a time also. In WOS-ELM, each sample can update the model without any delay. Its performance is compared with the OTER method [13]. OTER learns one sample at a time and use fixed weights to minimize the TER. OTER is an effective online learning method, especially if the class sizes are different. As can be observed from Table 7, on 13 out of 15 datasets, WOS-ELM achieves higher G_{mean} than OTER. A comparison of G_{mean} values obtained by WOS-ELM and OTER is shown in Fig. 2.

Training time of WOS-ELM is higher than OTER due to the tuning of weights upon the arrival of minority class samples. Since only one minority sample is learnt at a time, the tuning introduces a very small delay in each model update. No tuning is required upon the arrival of majority class samples since the weight is fixed at 1.

For chunk-by-chunk learning (see Table 5 and 6), no significant difference between WOS-ELM and OS-ELM-SOMTE in terms of generalization performance is observed. However, in terms of computational time WOS-ELM is preferred over OS-ELM-SOMTE. WOS-ELM clearly outperformed OS-ELM-UNDER on all the datasets used.

For one-by-one learning (see Table 7), the importance of weight tuning is validated. WOS-ELM obtains higher G_{mean} than OTER on almost all the datasets. OTER tries to optimize TER using fixed weights which may not be optimal for imbalanced datasets, while WOS-ELM tune weights to optimize G_{mean}, an appropriate CIL performance measure.

4.4 Statistical Significance Test

In order to analyze whether WOS-ELM is statistically significant than the other online sequential algorithms in comparison, we used Friedman test [32] for both chunk-by-chunk and one-

Table 7 Classification results comparison for one-by-one learning

Dataset	Algorithm	G_{mean}	Training time (s)
Page-block5	WOS-ELM	**91.7**	52.224
	OTER	91.5	24.351
Abalone15	WOS-ELM	**70.8**	5.854
	OTER	69.9	2.972
Yeast5	WOS-ELM	**80.2**	10.853
	OTER	79.1	7.85
car3	WOS-ELM	**94.8**	27.7
	OTER	93.1	20.014
Oil	WOS-ELM	**79.6**	7.268
	OTER	78.1	5.9
Abalone9-18	WOS-ELM	**84.4**	5.636
	OTER	82.8	3.127
Balance2	WOS-ELM	**67.6**	3.447
	OTER	66.8	2.103
Satimage4	WOS-ELM	**85.7**	197
	OTER	85.3	92.676
mf-morph10	WOS-ELM	**92.3**	5.298
	OTER	92.1	1.417
mf-zernike10	WOS-ELM	**87.7**	5.664
	OTER	86.5	1.592
cmc2	WOS-ELM	66	35.671
	OTER	**66.2**	11.529
Transfusion	WOS-ELM	**67.9**	1.071
	OTER	67.5	0.57
Haberman	WOS-ELM	**63.7**	0.558
	OTER	63.4	0.339
Phoneme	WOS-ELM	**84.3**	239.843
	OTER	84.2	79.512
Pima	WOS-ELM	75.6	12.833
	OTER	**76**	2.661

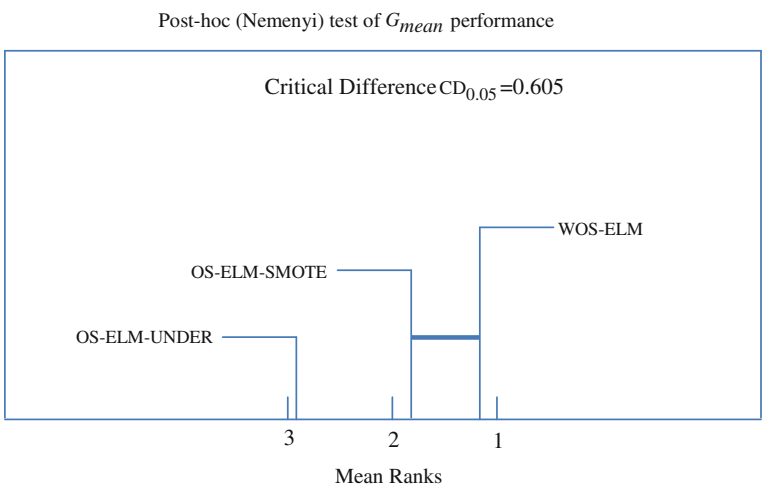

Fig. 3 Visualization of Nemenyi post-hoc test. The mean rank for each method over all the datasets is plotted on x-axis. Connected method are not significantly different at $p = 0.05$

by-one learning mode. For chunk-by-chunk mode, since three CIL methods are compared, a post-hoc analysis is also applied.

4.4.1 Chunk-by-Chunk Mode

WOS-ELM, OS-ELM-SMOTE and OS-ELM-UNDER are evaluated on 15 datasets with two different chunk sizes. Thus, a total of thirty test results are accumulated using each of the online methods. For significance testing, average ranks among the classifiers are calculated. For each dataset, the best performing algorithm is ranked as 1, the second best as 2, and so on. If classifiers produce same result, an average rank is assigned to each of them. We used *Matlab* function *'friedman'* for hypothesis testing. If p-value is lower than a specific confidence level α, null hypothesis is rejected (i.e. all the classifiers are equivalent and any differences among their average ranks are random). In our case, the null hypothesis is rejected at $\alpha = 0.05$ since the p-value for all G_{mean} is 6.6532×10^{-11}. Next a post-hoc Nemenyi test is adopted to observe the differences among the three classifiers. At a certain confidence level, a pair of classifiers is considered significantly different if the rank difference between them is larger than the critical difference (CD). A CD of 0.605 is obtained at $p = 0.05$. The average ranks of WOS-ELM, OS-ELM-SMOTE and OS-ELM-UNDER are 1.23, 1.83 and 2.96 respectively. The rank difference between WOS-ELM and OS-ELM-SMOTE is marginally lower (0.6) than the CD (0.605) so they are considered statistically similar. OS-ELM-UNDER is found to be statistically different (inferior) from the other two methods. Fig. 3 visualizes the Nemenyi post-hoc test at $p = 0.05$. Statistically similar methods are grouped with horizontal bar [32]. The mean ranks over the datasets for each method are shown in descending rank order on the horizontal axis. Since OS-ELM-SMOTE also performed better than OS-ELM-UNDER on almost all the datasets, the statistical significance test is unable to detect the difference between OS-ELM-SMOTE and WOS-ELM, despite WOS-ELM wining on most of the datasets.

4.4.2 One-by-One Mode

In this mode, 15 test results are accumulated from both WOS-ELM and OTER. At a confidence level of 0.05, null hypothesis is rejected since obtained p-value is 0.0045. Thus, WOS-ELM is statistically better than OTER at a confidence level of 0.05.

4.5 Summary of Experimental Results and Discussion

WOS-ELM is shown to have comparable or better verification performance than existing online sequential algorithms. It outperforms OS-ELM-UNDER on all the datasets. As observed from Fig. 1, in chunk-by-chunk mode, WOS-ELM and OS-ELM-SMOTE obtain close results with WOS-ELM winning on most of the datasets. The training time of WOS-ELM is found to be lower than the competing data sampling methods.

In one-by-one mode, WOS-ELM obtains better performance than OTER on most of the datasets, as shown in Fig. 2. This improvement is due to adopting the weight tuning instead of using the fixed weights by OTER.

The statistical significance test shows that WOS-ELM is better than OS-ELM-UNDER and similar to OS-ELM-SMOTE in terms of G_{mean} at $\alpha = 0.05$. WOS-ELM is also found to be statistically different from (superior to) the OTER method in terms of G_{mean} at $\alpha = 0.05$.

WOS-ELM is a flexible CIL algorithm that can learn data one-by-one or chunk-by-chunk with any chunk size. Unlike data sampling methods, each sample can update the model without depending on other samples from the same or the different class. It does not involve

any extra data processing cost such as finding k-NN or performing clustering. Thus, it is generally faster than the competing data sampling approaches.

5 Conclusion

In this paper, an online sequential method for class imbalance learning has been proposed. This method is referred to as weighted online sequential extreme learning machine (WOS-ELM). The new method is different from the typically used data sampling methods in that it can tackle the class imbalance problem in both chunk-by-chunk and one-by-one mode. Moreover, in the proposed WOS-ELM, previously learnt data is not required to be stored in the system. Even the old minority class samples can be discarded. Each arriving sample can update the model without any delay. Thus, the proposed method is better suited for real-time online applications. The computational time of WOS-ELM is also lower than those of the competing data sampling methods. Different from the fixed weight setting in the OTER method for TER minimization, the weights in WOS-ELM are tuned for G_{mean} optimization. An appropriate weight setting for CIL is selected in a computationally efficient manner. Faster training speed and comparable or better classification performance of the WOS-ELM algorithm than competing data sampling methods have been verified on 15 real-world imbalanced datasets. The proposed WOS-ELM is also seen to outperform OTER on most of the datasets in terms of G_{mean}.

Acknowledgment The authors would like to thank the anonymous reviewers whose insightful and helpful comments greatly improved this paper.

References

1. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
2. Haibo H, Garcia E (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
3. Kubat M, Matwin S (1997) Addressing the curse of imbalanced training sets: one-sided selection. Int Conf Mach Learn, Nashville, TN
4. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357
5. Ling CX, Sheng VS (2008) Cost-sensitive learning and the class imbalance problem. In: Sammut C (ed) Encyclopedia of machine learning. Springer, Berlin
6. Ozawa S, Pang S, Kasabov N (2008) Incremental learning of chunk data for online pattern classification. IEEE Trans Neural Netw 19(6):1061–1074
7. Lian NY, Huang GB, Saratchandran P, Sundararajan N (2006) A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw 17(6):1411–1423
8. Huang GB, Saratchandran P, Sundararajan N (2004) An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. IEEE Trans Syst Man Cybern B Cybern 34(6):2284–2292
9. Huang GB, Saratchandran P, Sundararajan N (2005) A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Trans Neural Netw 16(1):57–67
10. Polikar R, Upda L, Upda SS, Honavar V (2001) Learn++: an incremental learning algorithm for supervised neural networks. IEEE Trans Syst Man Cybern C Hum Syst 31(4):497–508
11. Shilton A, Palaniswami M, Ralph D, Tsoi AC (2005) Incremental training of support vector machines. IEEE Trans Neural Netw 16(1):114–131
12. Muhlbaier M, Polikar R (2007) Multiple classifiers based incremental learning algorithm for learning nonstationary environments. International Conference on Machine Learning and Cybernetics, pp 3618–3623, Hong Kong, China.
13. Kim Y, Toh KA, Teoh ABJ, Eng HL, Yau WY (2013) An online learning network for biometric scores fusion. Neurocomputing 102:65–77

14. Ditzler G, Polikar R, Chawla NV (2010) An incremental learning algorithm for non-stationary environments and class imbalance. International Conference on, Pattern Recognition, pp 2997–3000.
15. Ditzler G, Muhlbaier M, Polikar R (2010) Incremental learning of new classes in unbalanced datasets: learn++. UDNC. MCS Lect Notes Comput Sci 5997:33–42
16. Gao J, Fan W, Han J, Yu PS (2007) A general framework for mining concept-drifting streams with skewed distribution. SIAM International Conference on Data Mining, vol. 7.
17. Chen S, He H (2009) SERA: selectively recursive approach towards nonstationary imbalanced stream data mining. International Joint Conference on Neural Networks, Atlanta
18. Chen S, He H (2011) Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. Evolv Syst 2(1):35–50
19. Wang Y, Zhang Y, Wang Y (2009) Mining data streams with skewed distribution by static classifier ensemble. Stud Comput Intell 214:65–71
20. Nguyen H, Cooper E, Kamei K (2011) Online learning from imbalanced sata dtreams. International Conference of Soft Computing and Pattern Recognition (SoCPaR), pp 347–352.
21. Toh KA, Eng HL (2008) Between classification-error approximation and weighted least-squares learning. IEEE Trans Pattern Anal Mach Intell 30(4):658–669
22. Toh KA (2008) Deterministic neural classification. Neural Comput 20(6):1565–1595
23. Tang Y, Zhang Y, Chawla NV, Krasser S (2009) SVMs modeling for highly imbalanced classification. IEEE Trans Syst Man Cybern B Cybern 39(1):281–288
24. Akbani JR, Kwek S, Japkowicz N (2004) Applying support vector machines to imbalanced datasets. 15th European Conference on Machine Learning, Pisa, Italy, pp 39–50.
25. Batuwita R, Palade V (2010) FSVM-CIL: fuzzy support vector machines for class imbalance learning. IEEE Trans Fuzzy Syst 18(3):558–571
26. Batuwita R, Palade V (2012) Adjusted geometric-mean: a novel performance measure for imbalanced bioinformatics datasets learning. J Bioinf Comput Biol 10(4):23. doi:10.1142/S0219720012500035
27. Batuwita R, Palade V (2012) Class imbalance learning methods for support vector machines. In: Haibo He, Yunqian Ma (eds) Imbalanced learning: foundations, algorithms, and applications, Wiley, (in press).
28. Batuwita R, Palade V (2009) AGm: a New performance measure for class imbalance learning. Application to bioinformatics problems. The 8th IEEE International Conference on Machine Learning and Applications, Miami, USA, pp 545–550.
29. Hoens TR, Chawla NV (2012) Learning in non-stationary environments with class imbalance. International conference on Knowledge discovery and data mining, pp 168–176.
30. Frank A, Asuncion A (2010) UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml. Accessed on 10 June 2012
31. Kubat M, Holte RC, Matwin S (1998) Machine learning for the detection of oil spills in satellite radar images. Mach Learn 30:195–215
32. Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30