Neurocomputing 149 (2015) 316–329

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift

Bilal Mirza[a,*], Zhiping Lin[a], Nan Liu[b]

[a] School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore
[b] Department of Emergency Medicine, Singapore General Hospital, Singapore 169608, Singapore

ARTICLE INFO

Article history:
Received 19 August 2013
Received in revised form
20 January 2014
Accepted 19 March 2014
Available online 9 September 2014

Keywords:
Class imbalance
Concept drift
Extreme learning machine
Online learning
Recurring environments

ABSTRACT

In this paper, a computationally efficient framework, referred to as ensemble of subset online sequential extreme learning machine (ESOS-ELM), is proposed for class imbalance learning from a concept-drifting data stream. The proposed framework comprises a main ensemble representing short-term memory, an information storage module representing long-term memory and a change detection mechanism to promptly detect concept drifts. In the main ensemble of ESOS-ELM, each OS-ELM network is trained with a balanced subset of the data stream. Using ELM theory, a computationally efficient storage scheme is proposed to leverage the prior knowledge of recurring concepts. A distinctive feature of ESOS-ELM is that it can learn from new samples sequentially in both the chunk-by-chunk and one-by-one modes. ESOS-ELM can also be effectively applied to imbalanced data without concept drift. On most of the datasets used in our experiments, ESOS-ELM performs better than the state-of-the-art methods for both stationary and non-stationary environments.

1. Introduction

The class imbalance problem has been studied extensively [1–5] in the past decade or so. Imbalanced datasets are common in real-world applications such as medical diagnosis, fraud detection, spam filtering, bioinformatics, text classification, etc [1]. Recently, the class imbalance problem in sequential learning has also attracted attention of researchers from various application domains [6–9].

Sequential or incremental learners are computationally efficient as compared to batch learners [10–13] since batch learners require retraining with the complete dataset whenever new samples arrive. Sequential learners store the previously learnt information and update themselves only with the newly arrived data samples. However, since statistical characteristics of training data streams may change over time, class concepts tend to drift in non-stationary environments. Concept drift learning raises the so-called stability-plasticity dilemma. Sequential learning framework should be able to achieve a meaningful balance between previously acquired information (stability) and learning new information (plasticity) [12]. The class imbalance problem further complicates sequential learning from drifting data streams.

Class imbalance and concept drift are two challenging problems which can occur in the same data stream. Recently, class imbalance problem in drifting data streams has received attention for chunk-by-chunk learning [14–19]. Learn^{++}.NIE [16] is a state-of-the-art method in this area. Learn^{++} family of methods is based on ensemble learning framework in which a new classifier is trained with each arriving chunk of data and added to the ensemble. Learn^{++}.NIE works better than most of the competing methods in recurring environments since it does not remove old classifiers from the ensemble. For imbalance datasets, the prior knowledge of recurring concepts can be particularly useful since the minority class samples are usually rare. In general, incremental learning methods should meet the single-pass requirement i.e. once samples are learnt, they are discarded [12]. However, some methods store previously learnt minority class samples, thus, violating the single-pass requirement [17–19]. Gao [17] proposed to collect the minority class samples from all previous chunks while SERA [18] and REA [19] select the minority class samples from previous chunks which are similar to samples in the current chunk. Moreover, most of the existing methods assume that a full chunk of data is always available for training [14]. If the samples are arriving continuously or one-by-one, updating of the classification model is delayed until a full chunk is completed. Hence, the need for a class imbalance learning (CIL) method for non-stationary environments, which can learn in both chunk-by-chunk and one-by-one modes, is timely.

Extreme learning machine (ELM) [11,20] is becoming popular in large dataset and online learning applications due to its fast learning speed. ELM provides a single step least square estimatation (LSE) method for training single hidden layer feed forward network (SLFN)

* Corresponding author.
E-mail addresses: bilal2@e.ntu.edu.sg (B. Mirza), ezplin@ntu.edu.sg (Z. Lin), liu.nan@sgh.com.sg (N. Liu).

http://dx.doi.org/10.1016/j.neucom.2014.03.075
0925-2312/© 2014 Elsevier B.V. All rights reserved.

instead of using iterative gradient descent methods, such as back propagation algorithms. Very recently, a weighted online sequential extreme learning machine (WOS-ELM) was proposed for class imbalance learning [9]. WOS-ELM has been shown to effectively tackle the class imbalance problem in both chunk-by-chunk and one-by-one learning. However, WOS-ELM was proposed only for stationary environments and may not be appropriate for concept drift learning. Moreover, OS-ELM [11] related methods, with random hidden node parameters, may not always adapt well to the new data [21]. Thus, ensemble methods [21–23] are generally preferred over single OS-ELM methods [6,9,11].

In this paper, a computationally efficient framework, referred to as ensemble of subset online sequential extreme learning machine (ESOS-ELM), is proposed for class imbalance learning from a concept-drifting data stream. In ESOS-ELM, a minority class sample is processed by 'm' classifiers ('m' is the imbalance ratio) while a majority class sample is processed by a single classifier. The majority class samples are processed in a round robin fashion, i.e., the first majority class sample is processed by the first classifier, the second sample by the second classifier and so on. In this way, classifiers in the ensemble are trained with balanced subsets from the original imbalanced dataset. Note that the proposed framework tackles class imbalance and concept drift problems in both the one-by-one and chunk-by-chunk modes.

Ensemble learning methods are widely used in concept drift learning. Compared to single classifier methods, ensemble methods tend to better cope with concept drift problem, particularly with gradual drifts [14]. Dynamic weighted majority (DWM) [24] is a state-of-the-art ensemble method for concept drift learning with balanced datasets. In DWM, voting weights are decreased proportional to the error rate of the classifier. ESOS-ELM also uses dynamic weighted majority voting for concept drift learning. For tackling the class imbalance problem, ESOS-ELM processes incoming samples in a way that each OS-ELM network is trained with approximately equal number of minority and majority class samples. Unlike DWM, voting weights are updated proportional to an appropriate performance measure for CIL. In recurring environments [12,16,25], DWM may not be able to leverage the prior knowledge since old concepts are usually forgotten. To avoid this problem, we propose a novel information storage mechanism, using ELM theory, which is efficient both in terms of memory and computation. A change detection mechanism is also employed in the learning framework to promptly react to both the sudden and gradual drifts.

The new framework achieves better performance than that of Learn^{++}.NIE(gm) on most of the datasets used in this paper. ESOS-ELM achieves this performance with fewer hypotheses than in Learn^{++}.NIE(gm). The new method also obtained better performance than DWM in recurring environments. This superiority is due to the ELM-Store module which helps leverage the prior knowledge of old concepts. ESOS-ELM is also applied on benchmark imbalanced datasets without concept drift. ESOS-ELM outperformed WOS-ELM, OTER and SMOTE for all the 15 imbalanced datasets used in [9].

This paper is organized as follows: Section 2 discusses the preliminaries. Section 3 presents the details of the ESOS-ELM method. This is followed by experiments for validating the performance of the proposed framework in Section 4. Finally, Section 5 concludes the paper.

2. Preliminaries

2.1. ELM and OS-ELM

Extreme learning machine (ELM) [20] is a single step least square error estimate solution originally proposed for single hidden layer feed forward networks and later extended for non-neuron like networks. The input weights and biases connecting input layer to the hidden layer (hidden node parameters) are assigned randomly and the weights connecting the hidden layer and the output layer (output weights) are determined analytically. Compared to the traditional iterative gradient decent methods such as back propagation algorithm, training is extremely fast in ELM.

Consider a training dataset with q classes $\{x_i, y_i\}$, $i = 1, \ldots.., N$ and $y_i \in R^q$. $x_i \in R^d$ is a d-dimensional data point. If $G(x)$ is an infinitely differentiable activation function such as sigmoidal function in the hidden layer, the output of SLFN is given by

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

where L is the number of hidden nodes, a_j and b_j are the jth hidden node's learning parameters assigned randomly, $j = 1, \ldots, L$. $\beta_j \in R^q$ is the output weight vector connecting the jth hidden node to the output nodes. The SLFN output can be expressed in the matrix form as below

$$O = H\beta$$
$$\text{where} \qquad H = \begin{bmatrix} G(a_1, b_1, x_1) & \ldots & G(a_L, b_L, x_1) \\ \vdots & \ldots & \vdots \\ G(a_1, b_1, x_N) & \ldots & G(a_L, b_L, x_N) \end{bmatrix}_{N\times L} \tag{2}$$

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 $O = [o_1, o_2, \ldots, o_N]^T$. A LSE solution of (2), referred to as extreme learning machine [20], can be obtained as

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

where $Y = [y_1, y_2, \ldots, y_N]^T$ and $H^{\dagger}$ is the Moore–Penrose generalized inverse of matrix H. The above output weight matrix β minimizes the cost function $\|O - Y\|$.

ELM has been successfully applied to real-time online applications due to its fast learning speed and good generalization performance. Online sequential extreme learning machine (OS-ELM) is becoming more and more popular in online and huge datasets applications [11]. Training in OS-ELM consists of two phases i.e. initialization and sequential learning. In the initialization phase, a small portion of training data $n_0 = \{x_i, y_i\}$, $i = 1, \ldots\ldots, N_0$ is considered. The initial output weight matrix is calculated according to the ELM algorithm as below

$$\beta^{(0)} = P_0 H_0^T Y_0$$
$$\text{where} \qquad P_0 = (H_0^T H_0)^{-1} \tag{4}$$

Upon the arrival of $(k+1)$th chunk, the partial hidden layer output matrix H_{k+1} is computed first and then the output weight matrix is updated according to the following recursive least squares algorithm.

$$P_{k+1} = P_k - P_k H_{k+1}^T (I + H_{k+1} P_k H_{k+1}^T)^{-1} H_{k+1} P_k$$
$$\beta^{(k+1)} = \beta^{(k)} + P_{k+1} H_{k+1}^T (Y_{k+1} - H_{k+1} \beta^{(k)}) \tag{5}$$

The above equation is valid for any chunk size, including one sample at a time learning. For detailed derivation of (5) readers can refer to [11].

2.2. WELM and WOS-ELM

Recently, weighted extreme learning machine (WELM) has been proposed as a cost-sensitive algorithm for class imbalance learning [5,9]. In [5], fixed weights are assigned to the samples proportional to their respective class sizes. Although the proposed fixed weights suggested in [5] improve the classification performance for imbalanced datasets, they may not be optimal.

The output matrix for WELM is given below [9]

$$\beta=(H^TWH)^{-1}H^TWY \tag{6}$$

where $W=W_-+W_+$ and $W_-=\text{diag}\big((1/m^-),\dots,(1/m^-),0,\dots,0\big)$ $W_+=\text{diag}\big(0,\dots,0,(1/m^+),\dots,(1/m^+)\big)$

m^- and m^+ are sizes of the positive and negative classes. Note that the elements of H and Y are ordered according to the classes of the samples i.e. all the negative class samples come first, followed by the positive class samples. An informative weight tuning is performed to optimize weights for *Gmean* in [9]. *Gmean* is a commonly used evaluation measure for CIL. It measures the true positive rate (*TPR*) and the true negative rate (*TNR*) in a balanced way as follows

$$Gmean=\sqrt{TPR\times TNR}$$

Gmean is used when combined performance of the two classes needs to be judged by a single metric in imbalanced environments. For highly imbalanced datasets, a high overall accuracy can always be achieved if *TNR* is very high even with *TPR* equals to zero. On the other hand, a high *Gmean* value requires both *TPR* and *TNR* to be high. The weight settings used for *Gmean* optimization are (m^-,m^+), $(1.1m^-,m^+)$, $(1.2m^-,m^+)$,.., $(2m^-,m^+)$, $(m^-,1.1m^+)$, $(m^-,1.2m^+)$,.., $(m^-,2m^+)$. An online sequential version of WELM is also proposed in [9]. Similar to OS-ELM, the learning by a weighted online sequential extreme learning machine (WOS-ELM) consists of two steps as follows

Step1: Initialization

The weighted least square solution corresponding to the initialization chunk n_0 is given by

$$\beta^{(0)}=K_0^{-1}H_0^TW_0Y_0 \tag{7}$$

$$K_0=H_0^TW_0H_0 \tag{8}$$

The weighting matrix W_0 consists of $w_0^+=(m_0^-/m_0^+)$ and $w_0^-=1$. The weighting matrix is tuned to optimize *Gmean* e.g. *TPR* can be increased using $1.2w_0^+$, $1.5w_0^+$, $1.8w_0^+$, $2w_0^+$ as an attempt to increase *Gmean*.

Step 2: Sequential learning

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

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

$$K_1=\begin{bmatrix}H_0\\H_1\end{bmatrix}^T\begin{bmatrix}W_0&0\\0&W_1\end{bmatrix}\begin{bmatrix}H_0\\H_1\end{bmatrix} \tag{9}$$

The updating equation for WOS-ELM is given below

$$P_{k+1}=P_k-P_kH^T{}_{k+1}(W_{k+1}^{-1}+H_{k+1}P_kH^T{}_{k+1})^{-1}H_{k+1}P_k$$
$$\beta^{(k+1)}=\beta^{(k)}+P_{k+1}H^T{}_{k+1}W_{k+1}\left(Y_{k+1}-H_{k+1}\beta^{(k)}\right) \tag{10}$$

where $P_{k+1}=K_{k+1}^{-1}$.

Every time a new chunk of samples arrive, W_{k+1} is tuned to maximize *Gmean*. Note that when the size of the chunk of samples is equal to 1, the chunk-by-chunk learning becomes one-by-one learning.

3. Method

In this section, an ensemble of subset online sequential extreme learning machine (ESOS-ELM) is proposed for class imbalance learning from drifting data stream. As shown in Fig. 1, the proposed ESOS-ELM method consists of three blocks, the main ensemble block, the ELM-Store block and the change detector block. These blocks are discussed in details as follows.

3.1. The main ensemble

Similar to the ensemble of OS-ELM (EOS-ELM) method [21], we initialize several OS-ELM networks in ESOS-ELM, with different random hidden node parameters and the same number of hidden neurons. Some OS-ELM networks adapt faster and better to new data than others. The ensemble output generally provides a more stable solution than that of a single OS-ELM network [21]. EOS-ELM is only for balanced datasets while ESOS-ELM can overcome the class imbalance problem. In the new method, each minority class sample is processed by '*m*' classifiers ('*m*' is the imbalance ratio) while the majority class samples are processed in a round robin fashion i.e. first majority class sample is processed by the first classifier, the second sample by the second classifier and so on. Classifiers in ESOS-ELM are trained with approximately equal number of majority and minority class samples. Instead of training every OS-LEM network with all the samples, they are trained with balanced subsets of the data stream.

Depending on whether the imbalance ratio is moderate or high, ESOS-ELM works slightly differently. In this paper, an imbalance ratio that is equal to or smaller than the ensemble size is considered moderate while an imbalance ratio greater than the ensemble size is considered high. ESOS-ELM maintains two sorted lists of classifiers in ascending order, the first list is sorted with respect to the number of minority class samples processed by each classifier while the second list with respect to the number of majority class samples. If the imbalance ratio is smaller than the ensemble size, the incoming minority class sample(s) is processed by the top '*m*' classifiers in the first sorted list. While the majority

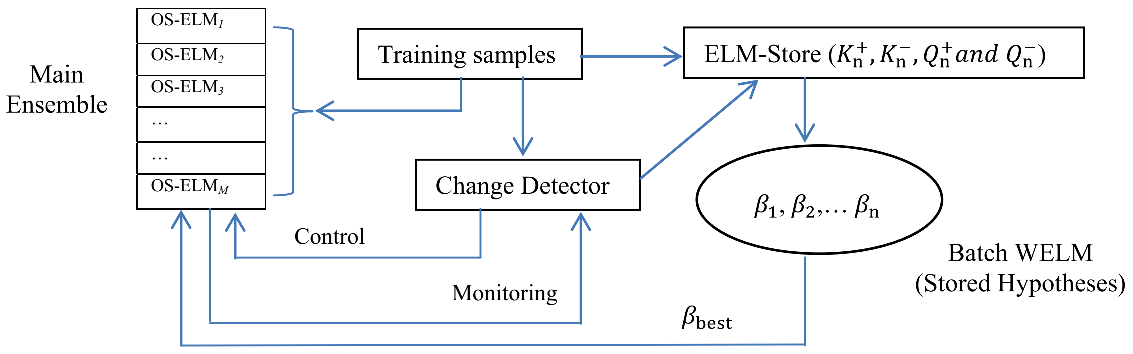

Fig. 1. ESOS-ELM for non-stationary recurring environments.

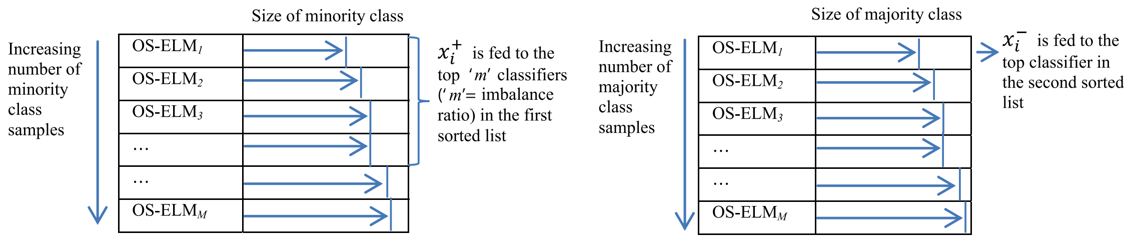
Size of minority class
Increasing number of minority class samples
OS-ELM$_1$
OS-ELM$_2$
OS-ELM$_3$
…
…
OS-ELM$_M$
x_i^+ is fed to the top 'm' classifiers ('m'= imbalance ratio) in the first sorted list
Size of majority class
Increasing number of majority class samples
x_i^- is fed to the top classifier in the second sorted list

Fig. 2. One-by-one sequential learning in ESOS-ELM for imbalanced data streams.

class samples are distributed in a round robin fashion, starting with the top classifier in the second sorted list. Assume that the imbalance ratio is 5. For one-by-one learning, the minority class sample is processed by top 5 classifiers in the first sorted list while the majority class sample is processed by the top classifier in the second sorted list (see Fig. 2). The two sorted lists are updated after each time step.

For chunk-by-chunk learning, suppose that the new chunk contains 50 minority class samples and 250 majority class samples. All the minority class samples are processed by the top 5 classifiers in the first sorted list, while 50 majority class samples are processed by 5 classifiers in a round robin fashion i.e. the first 50 samples are processed by the first classifier in the second sorted list, the next 50 samples by the second classifier and so on. Even though the original dataset is imbalanced, classifiers in the ensemble are trained with similar number of majority and minority class samples.

If the imbalance ratio is high, i.e. greater than the ensemble size, the arriving minority class sample(s) is processed by all the classifiers. The majority class samples can be processed similarly to the moderate imbalance ratio case. However, for high imbalance ratios e.g. 100, classifiers are still trained with imbalanced subsets with a lower imbalance ratio of 10, if the ensemble size is 10. In such cases, we only select the majority class samples for training which are misclassified by the ensemble.

Dynamic weighted majority (DWM) [24] is a state-of-the-art ensemble method for concept drift learning. However, it has some limitations when used for imbalanced data streams. Firstly, every classifier in the ensemble will be biased towards the majority class since they are trained with imbalanced dataset. Secondly, voting weights are updated with the assumption that errors made by the two classes are equal i.e. DWM does not consider imbalance in the dataset for its weight updating. In ESOS-ELM, we also use dynamic weighted majority voting for concept drift learning. Existing classifiers in ESOS-ELM are updated with new data according to the balanced subset approach discussed earlier and shown in Fig. 2. The output matrices are updated using (5). If the current ensemble misclassifies a sample, a new classifier is added using (6) to make sure it is not biased towards the majority class. Classifiers can be added and removed at every time step or after every "*p*" time steps. For imbalanced datasets, the overall accuracy or total error rate is not an appropriate evaluation measure. If the voting weight is updated considering error from both classes as equal, the ensemble may not be able to promptly detect changes in the minority class. Therefore, voting weights should also be updated differently than the weights in original DWM. In ESOS-ELM, classifier weight is multiplied by its *Gmean* value on the current environment. As discussed earlier, *Gmean* is a commonly used evaluation criterion for CIL which measures the accuracy of both the classes in a balanced way [26–28]. The weights are normalized after each learning step so that the highest weight is always 1.

DWM has a potential of adding large number of classifiers, particularly for difficult classification tasks [24]. With constrained computational and memory resources, a limit can be set on the number of classifiers. If the limit is reached, classifier with the lowest weight is replaced with a new one. It was pointed out that placing a limit on the number of classifiers may not have appreciable performance impact [24]. In ESOS-ELM, we also set a limit of 15 classifiers on the ensemble size. However, old concepts which are not relevant now can be forgotten since classifiers trained with old concepts may be removed over time. With cyclic or recurring environments [12,16,25] i.e. when old class concepts recur after a long period, DWM usually starts learning concepts from the beginning since it no longer has access to the forgotten information. Prior knowledge of recurring concepts is particularly useful in class imbalance learning since the minority class samples can be rare. Using extreme learning machine theory, we propose a computationally efficient information storage scheme to retain old information. We call this storage scheme as ELM-Store which is discussed as follows.

3.2. ELM-Store for saving old information

In order to leverage the prior knowledge of recurring concepts, previously learnt information should be retained in an efficient way. Adaptive Classifiers-Ensemble systems (ACE) [25] uses an online classifier trained on the current concept (short-term memory) and many batch classifiers for retaining old concepts (long-term memory). Classifiers outputs are combined using weighted majority voting. It uses a long-term buffer to store the most recent samples. A new batch classifier is trained whenever concept drift is detected or the buffer is full. This approach to storing prior information may not be very efficient in terms of both memory and computation. If the long-term buffer is too big, we require a big memory to store old samples. The size of memory may be a constraint in various applications. It also violates the single-pass requirement of incremental learners. If the buffer size is too small, the number of stored hypotheses or batch classifiers grows very fast as ACE does not perform any pruning. In this case, the learning framework becomes computationally very expensive. Using ELM theory, we propose an efficient way to retain old information using batch classifiers. These batch classifiers constitute the long-term memory while the main ensemble, described in the previous subsection represents the current or the short-term memory. Weighted ELM (WELM) is used as the batch classifier for CIL [5,9]. The information acquired from training samples is accumulated independent of any long-term buffer. A batch classifier is only trained and stored whenever ensemble performance falls below a performance threshold which usually reflects concept drift. Thus, the number of batch classifiers does not grow rapidly. If any stored hypothesis performs better than the current ensemble, indicating possible recurring concept, it is introduced in the ensemble with a weight of 1. We observe from the experiments that stored hypotheses obtaining *Gmean* greater than 95% of the main ensemble *Gmean* help improve the ensemble performance. This is in contrast to DWM which introduces new classifiers trained on the current data and does not make use of the prior knowledge. We hope to train and store just one (or at most a few) hypothesis per concept, which has the exact performance of a batch learner (e.g. WELM), while computational time and memory requirement are the same or lower than that of an incremental learner (e.g. WOS-ELM). Consider the output weight equation of ELM in (3).

$$\beta = H^{\dagger} Y = (H^T H)^{-1} H^T Y \tag{11}$$

At time step $t=0$, let

$$K_0 = H_0^T H_0$$

where H_0 is the initial hidden layer output matrix calculated using initialization set n_0, as in (4). At the subsequent time step:

$$K_1 = K_0 + H_1^T H_1$$

where H_1 is the hidden layer output matrix calculated with new data arriving at the current time step.Generalizing to time step 'n' ($n=1,2,\ldots$):

$$K_n = K_{n-1} + H_n^T H_n$$

Similarly,

$$Q_0 = H_0^T Y_0$$

Then

$$Q_1 = Q_0 + H_1^T Y_1$$
$$\vdots$$
$$Q_n = Q_{n-1} + H_n^T Y_n$$

Thus (11) can be written as

$$\beta = K_n^{-1} Q_n$$

where $K_n \in R^{LXL}$, $Q_n \in R^{LXq}$, L is the number of hidden neurons and q represents the number of classes. In this paper, we only consider two class problems i.e. $q=2$. Note that K_n and Q_n act as buffers for storing information from training samples similar to the long-term buffer in ACE. However, different from ACE's buffer, the sizes of K_n and Q_n are fixed and dependent on the number of hidden neurons rather than the number of samples. Splitting K_n and Q_n according to positive and negative classes:

$$K_n = K_n^+ + K_n^-$$
$$K_n^+ = K_{n-1}^+ + (H_n^+)^T H_n^+$$
$$K_n^- = K_{n-1}^- + (H_n^-)^T H_n^-$$
$$Q_n = Q_n^+ + Q_n^-$$
$$Q_n^+ = Q_{n-1}^+ + (H_n^+)^T Y_n^+$$
$$Q_n^- = Q_{n-1}^- + (H_n^-)^T Y_n^-$$

For CIL, we use WELM (6) instead of ELM as below

$$\beta = (H^T W H)^{-1} H^T W Y \tag{12}$$

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

m^- and m^+ are negative and positive sizes, respectively. Thus K_n and Q_n matrices for WELM becomes

$$K_n = \frac{1}{m^+} K_n^+ + \frac{1}{m^-} K_n^-$$
$$Q_n = \frac{1}{m^+} Q_n^+ + \frac{1}{m^-} Q_n^- \tag{13}$$

K_n^+ and $K_n^- \in R^{LXL}$, Q_n^+ *and* $Q_n^- \in R^{LX2}$. K_n^+, K_n^-, Q_n^+ *and* Q_n^- are collectively referred to as temporary matrices and they represent the ELM-Store.

When a concept drift is detected, a batch classifier (WELM) using (12) needs to be trained. Thus, we need to constantly update K_n^+, K_n^-, Q_n^+ and Q_n^-, whenever new sample(s) arrive. These temporary matrices have the dimensions dependent on the number of hidden neurons and not the number of samples, i.e., even after one million samples, the sizes of these temporary matrices still stay the same. Thus, we can store the information from all the samples using temporary matrices without any long-term buffer or extra memory. Also, the number of WELMs or stored hypotheses increases proportionally with the number of times the concept drift is detected and not with the number of arriving samples [25]. For example, if the concept drift occurs after one million samples belonging to the same concept, a single WELM (sudden drift) or a few WELMs (gradual drift) are trained, with incrementally updating temporary matrices. For such huge data streams, a large number of batch classifiers will be trained and stored in ACE.

In ELM, matrix inverse is usually the most expensive part in terms of computation. It is only performed once, i.e. when a concept drift is detected. Then K_n^+, K_n^-, Q_n^+ and Q_n^- are updated in an incremental way. Samples are discarded once the matrices are updated thus still meeting the single-pass requirement. Whenever a new sample or a chunk of samples arrives, at most four matrix additions and multiplications are performed in ELM-Store. Once the concept drift is detected and new β is calculated using (12), K_n^+, K_n^-, Q_n^+ and Q_n^- are reset and initialized with the best hidden node parameters in the main ensemble. The counters m^- and m^+ are also reset. With the ELM-Store technique, we manage to retain information of past concepts without occupying considerable memory space. In short, ELM-Store has the unique feature of storing information with which classifiers can be trained later having:

- Computation and memory requirements of incremental learners (updating K_n^+, K_n^-, Q_n^+ and Q_n^-).
- Performance of batch learners ($\beta = K_n^{-1} Q_n$)

3.3. Change detector

In general, there are two types of concept drifts to be detected, sudden drifts and gradual drifts. Sudden drifts are usually easier to detect as compared to gradual ones. In the proposed method, if performance of the main ensemble falls below certain threshold 'τ', a new hypothesis (WELM) is trained using ELM-Store. 'τ' is set to 90% of the *Gmean* value from the last evaluation. To make the ensemble react rapidly to sudden drifts, all the existing classifiers stop voting. In order to avoid false alarms, these classifiers are still kept in the ensemble and continuously updated with new data. Any old classifier yielding better *Gmean* than any of the new classifiers can start voting again. Note that the new batch WELM can also enter the main ensemble in case of false alarm.

Gradual drift is generally not very apparent since the performance degrades slowly. It can also be mistaken as noise. In this paper, gradual drift is detected using statistical decision theory similar to [29]. Other change detection algorithms can also be used. DWM can handle gradual drifts quite effectively. In ESOS-ELM, the gradual drift detection method does not change the way ensemble works. It is only used to train WELM using ELM-Store. In stationary conditions, a performance evaluation criterion G remains nearly constant. It is assumed that G follows Gaussian distribution. If the concept drifts gradually, G will move from one Gaussian to another Gaussian slowly. Hypothesis testing is used to detect this type of change. In hypothesis testing, there is a null hypothesis $\mathcal{H}_0$ and an alternate hypothesis $\mathcal{H}_1$. $\mathcal{H}_0$ corresponds to no change while $\mathcal{H}_1$ corresponds to a change. Suppose G_i is the *Gmean* of the current evaluation i and that the conditional probability density functions (*pdfs*) of G_i under $\mathcal{H}_0$ and $\mathcal{H}_1$ are $p(G_i|\mathcal{H}_0)$ and $p(G_i|\mathcal{H}_1)$, respectively. Once $p(G_i|\mathcal{H}_0)$ and $p(G_i|\mathcal{H}_1)$ are known, a likelihood ratio test can be performed as below

$$L(G_i) = \begin{cases} \mathcal{H}_0, & \frac{p(G_i|\mathcal{H}_1)}{p(G_i|\mathcal{H}_0)} < \tau \\ \mathcal{H}_1, & \frac{p(G_i|\mathcal{H}_1)}{p(G_i|\mathcal{H}_0)} \geq \tau \end{cases} \tag{14}$$

$L(G_i)$ greater than 'τ' corresponds to alternate hypothesis $\mathscr{H}_1$ which means change is detected. Otherwise there is no change detected. We split the sequence of *Gmean* values obtained from past evaluations into two parts. The first part represents the null hypothesis $\mathscr{H}_0$ with mean μ_0, variance σ_0^2 and *pdf* $p(G_i|\mathscr{H}_0)$. The second part represents the alternate hypothesis $\mathscr{H}_1$ with mean μ_1, variance σ_1^2 and *pdf* $p(G_i|\mathscr{H}_1)$. $p(G_i|\mathscr{H}_0)$ and $p(G_i|\mathscr{H}_1)$ can be estimated assuming Gaussian *pdf* as in (15) and (16).

$$p(G_i|\mathscr{H}_0) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(\frac{(G_i-\mu_0)^2}{2\sigma_0^2}\right) \tag{15}$$

$$p(G_i|\mathscr{H}_1) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(\frac{(G_i-\mu_1)^2}{2\sigma_1^2}\right) \tag{16}$$

Once $p(G_i|\mathscr{H}_0)$ and $p(G_i|\mathscr{H}_1)$ are estimated, the likelihood ratio test using (14) is performed. A smaller value of 'τ' can detect small changes but number of batch classifiers can grow very fast. If 'τ' is too high to detect a change, the algorithm will be the same as DWM i.e. ELM-Store has no contribution. *Gmean* values that are obtained by the new classifier (added at the time step when last change is detected) over subsequent time steps, form the sequence for calculating $p(G_i|\mathscr{H}_0)$ and $p(G_i|\mathscr{H}_1)$. The test using (14) is performed at every alternate evaluation step, starting with at least 6 previous evaluations. If there is no change, the next test is performed with additional *Gmean* values from the next two evaluations i.e. a sequence of past 8 *Gmean* values from the same classifier.

The main ensemble, ELM-Store and drift detection mechanism are combined and referred to as ESOS-ELM which is summarized in Algorithm 1 and also shown in Fig. 1. The samples are classified using weighted majority voting in the main ensemble. ESOS-ELM can promptly respond to both sudden and gradual drifts. The significance of ESOS-ELM is that it can tackle the class imbalance problem of concept-drifting data streams in both chunk-by-chunk and one-by-one learning. It provides a computationally and memory efficient alternative to the current state-of-the-art chunk-by-chunk learning methods. ESOS-ELM maintains fewer hypotheses to leverage prior knowledge of recurring concepts. Unlike various class imbalance methods [17–19], old minority class samples need not be stored.

It is of relevance to point out that ESOS-ELM can also be effectively applied to class imbalance problems without concept drift. Recently, WOS-ELM [5] and OTER [9] have been proposed to tackle the class imbalance problem in online sequential learning applications in stationary environments. WOS-ELM and OTER methods are modified versions of OS-ELM, thus, may not always adapt well to the new data due to the non-optimal hidden node parameters problem. Since ESOS-ELM is an ensemble method, it can provide a more stable solution [21]. In the experiments section, we will use ESOS-ELM to sequentially learn the benchmark imbalanced datasets and compare its results with those of the recently proposed methods.

Algorithm 1. ESOS-ELM training for moderate imbalance ratios

```
Given:
$M \rightarrow$ main ensemble size
$n_0 = n_0^+ + n_0^- \rightarrow$ Initialization set size
$x_i \rightarrow$ Training sample
$y_i \rightarrow$ Target label, $i \rightarrow 1$ to $n$
$p \rightarrow$ Interval between classifier addition, removal, weight
  update, and change detection test
$\tau \rightarrow$ Threshold for change detection
Output:
Hypotheses $\rightarrow \beta_u$
Hypotheses weight $w_u$, $u \rightarrow 1$ to $M$
Initialization
$ir_0 = \text{floor}(n_0^- / n_0^+)$, $c = \text{floor}(n_0^- / M)$
$a=1$, $b=c$, $d=1$, $e=c$, count$=0$
for u $\rightarrow$ 1,….,$M$
        Initialize OS-ELM using (4) $\rightarrow \beta_u(x_a^-, \ldots, x_b^-, x_d^+, \ldots x_e^+)$
        $w_u \rightarrow 1$
        $a=a+c$; $b=b+c$
        Count=count+1
        if count==$ir_0$
        $d=d+c$; $e=e+c$; count=0
        end
  end
  initialize ELM-store ($K_n^+$, $K_n^-$, $Q_n^+$ and $Q_n^-$)
Online sequential learning
    Index_p=sort_ascend ($\beta_u$,+)   %Sort classifiers with respect to positive class samples%
    Index_n=sort_ascend ($\beta_u$,−)   %Sort classifiers with respect to negative class samples%
    $O \rightarrow$ Ensemble_Classify ($x_i$)
    if (change_detection_test($\tau$)==1) %Change detected%
        train WELM using (13)
        Re-initialize ELM-Store
  else
        Update ELM-Store
  end
  if $y_i$=='+'
    for $u \rightarrow 1, \ldots, ir$ %minority class sample processed by $ir$ no. of classifiers ($ir \rightarrow$ imb. ratio)%
          $o \rightarrow$ Classify ($\beta_{Index\_p(u)}$, $x_i$)
          Weight_update ($o$, $y_i$,$p$)
          Update OS-ELM using (5) $\rightarrow \beta_{Index\_p(u)}(x_i)$
    end
  else
        $o \rightarrow$ Classify ($\beta_{Index\_n(1)}$, $x_i$)
        Weight_update ($o$,$y_i$,p)
        Update OS-ELM using (5) $\rightarrow \beta_{Index\_n(1)}(x_i)$ %majority class sample processed by single classifier%
  end
  if ($O^{\sim}=y_i$ and $i$ mod $p=0$)
    $M \rightarrow M+1$
    Initialize $\beta_M$ using (6)
        $w_M \rightarrow 1$
  End
```

Note: For high imbalance ratios, the algorithm is slightly different. A minority class sample is processed by all the classifiers while a majority class sample is processed by the top classifier in the negative sorted list, only if it is misclassified by the ensemble.

4. Experiments

The performance of ESOS-ELM is first evaluated for class imbalance learning with concept drift in Section 4.1. Later it is also evaluated for class imbalance learning without concept drift in Section 4.2.

4.1. Classification results on benchmark imbalanced datasets with concept drift

In order to show the effectiveness of ESOS-ELM in changing environments, four non-stationary datasets are used. For all the datasets, main ensemble is initialized with 10 classifiers and an upper limit of 15 is set on the ensemble size. ESOS-ELM has the distinction of learning in both one-by-one and chunk-by-chunk mode over existing methods. Hence, it is first compared with WOS-ELM for class imbalance learning in one-by-one mode. This comparison will show the effectiveness of the new CIL method over WOS-ELM in drifting datasets. Then it is compared with DWM approach to show the effectiveness of ELM-Store in recurring environments. Since DWM is originally for balanced classes, classifiers in DWM are trained in the same way as the classifiers in the main ensemble of ESOS-ELM (Fig. 2). This modified version of DWM is referred to as DWM-CIL. Note that we train DWM-CIL identical to ESOS-ELM for the initial part of the dataset and then observe difference between the two in the later part, when concepts recur. ESOS-ELM can make use of old knowledge through ELM-Store while DWM-CIL does not have such mechanism. Finally, in chunk-by-chunk mode, the new method is compared with Learn^{++}.NIE which is a state-of-the-art CIL method in non-stationary recurring environments. Details of datasets and classification results are given below.

4.1.1. The STAGGER concepts

The STAGGER concepts [30] is one of the widely used dataset for performance evaluation of classifiers in the presence of concept drift. The feature space is described by three attributes: size {small, medium, large}, color {red, green, blue} and shape {square, circular, triangle}. There are three data sources as below

- Target Concept 1: size = small AND color = red.
- Target Concept 2: color = green OR shape = circular.
- Target Concept 3: size = medium OR size = large.

We randomly generated 1200 instances belonging to one of the 3 concepts. The first 400 samples belong to Concept 1, the next 400 to Concept 2, and the last 400 to Concept 3. This is an example of sudden concept drift and class 1 is the target class. For Concept 1, imbalance ratio between the two classes is 7 while the remaining two concepts are fairly balanced. This is a very good example of varying imbalance ratios i.e. initially the classes are imbalanced but in subsequent concepts they become balanced. We repeated the same procedure to generate 1200 more instances in order to simulate recurring environment. In order to evaluate classifiers performance, 2400 instances are randomly generated for testing. Samples are learnt one-by-one. 'p' in this example is set to 40 i.e. new classifier can be added and old classifier with the lowest weight can be removed after every 40 samples. The number of hidden neurons is gradually increased and the one yielding the best overall *Gmean* is selected. The best number of hidden neurons is found to be 10. A limitation of the new method in one-by-one learning is that few samples should be available for initializing a new classifier using (6). It is preferred to have this number equals to or greater than the number of hidden neurons [11]. In order to meet the single-pass requirement, ELM-Store is used to incrementally store the information of recent most samples.

If the ensemble *Gmean* of the current environment is less than 90% of the last evaluation, all the old classifiers stop voting. This acts as a performance threshold. These classifiers are still updated with new data. Any old classifier achieving higher *Gmean* than any new classifier can start voting again. In this way, the new method can react quickly to the sudden drift. A batch classifier (WELM) is trained whenever overall *Gmean* of the ensemble falls below the performance threshold. This classifier can be added to the ensemble in case of false alarm since any stored WELM with a higher *Gmean* than that of the ensemble is added to the ensemble with a weight of 1. Figs. 3 and 4 compare the performance of ESOS-ELM with those of WOS-ELM and DWM-CIL, respectively. The average *Gmean* values over the entire dataset are reported in Table 1.

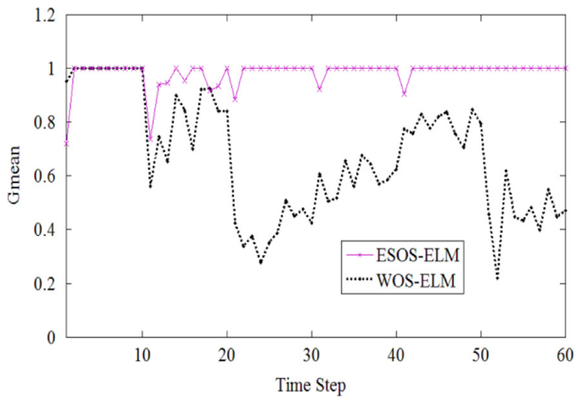

Fig. 3. ESOS-ELM and WOS-ELM comparison on STAGGER concepts.

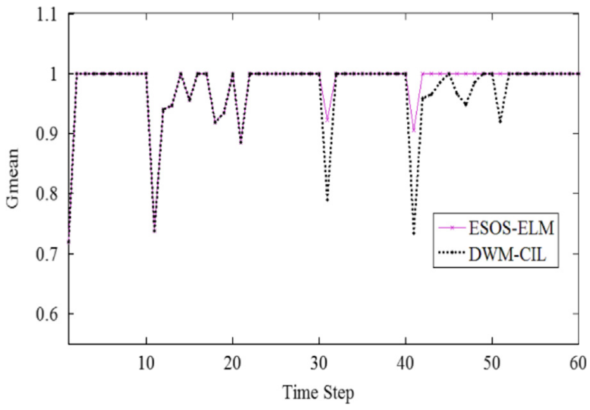

Fig. 4. ESOS-ELM and DWM-CIL comparison on STAGGER concepts.

Table 1
Classification results comparison for non-stationary environments (one-by-one).

Dataset	Algorithm	*Gmean*
STAGGER	DWM-CIL	97.2
	WOS-ELM	67.1
	ESOS-ELM	**98.1**
SEA	DWM-CIL	83.8
	WOS-ELM	82.6
	ESOS-ELM	**84.0**
Spiral	DWM-CIL	80.2
	WOS-ELM	52.8
	ESOS-ELM	**80.5**
Elec2	DWM-CIL	70.5
	WOS-ELM	65.6
	ESOS-ELM	**70.5**

Gmean values in the table are average of 5 trials. ESOS-ELM clearly outperformed WOS-ELM. The performance improvement over DWM-CIL (see time steps 30, 40 and 50 in Fig. 4) is due the fact that ESOS-ELM can make use of old knowledge of recurring concepts, using ELM-Store, which is completely or partially forgotten by DWM-CIL method. This improvement is achieved with storing 5 extra hypotheses (WELM).

4.1.2. The SEA concepts

The synthetic problem proposed by Street and Kim [31] has become a benchmark in the concept drift problem. The dataset consist of three attributes, two of them are relevant while the third one is noise. Values of the two relevant attributes (x_1 and x_2) are uniformly distributed between 0 and 10. The samples belong to class 1 if $x_1+x_2 \le b$ and class 2 otherwise, where b is the threshold. Similar to [16] threshold is varied from 4 to 7 twice to simulate recurring environment. Concept drift occurs after every 50,000 samples. This is also an example of sudden concept drift. In total, there are 200,000 samples for training and 200,000 for testing. Originally SEA does not have class imbalance problem so the target class is undersampled. The imbalance ratio is varied between 4 and 10. The class imbalance problem is overcome in one-by-one learning mode as shown in Fig. 2. The best number of hidden neurons is found to be 120. 'p' is set to 1000.

Whenever change is detected, a batch WELM is trained using ELM-Store. The performance threshold is set to 90% of *Gmean* from the last evaluation. Figs. 5 and 6 compare ESOS-ELM with WOS-ELM and DWM-CIL, respectively. The average *Gmean* values over the entire dataset are reported in Table 1. ESOS-ELM performs better than WOS-ELM. ESOS-ELM also performs slightly better than DWM-CIL due to ELM-Store. The table shows average *Gmean* values over entire period while the difference between ESOS-ELM and DWM-CIL is only in the second half of the entire period. The average *Gmean* values in the second half (when concepts re-occur) obtained by ESOS-ELM and DWM-CIL are 84.3 and 83.8, respectively. Whenever concepts drift in the second half, ESOS-ELM responds faster than DWM-CIL (see time steps 100–110 and 150–160 in Fig. 6) using ELM-Store. This gain is achieved by storing only 3 extra hypotheses (WELM).

4.1.3. Rotating spiral dataset

Rotating spiral is a recently introduced synthetic dataset [16,32] consisting of four rotating spirals, two for each class. The spirals rotate over 600,000 instances with environment recurring every 300,000 instances i.e. two complete rotations. Entire feature space is considered in the experiments. Rotating spirals change the decision boundary continuously thus simulating gradual drift. The best number of hidden neurons is found to be 120. Originally both classes have equal number of samples so class 1 is undersampled to achieve an imbalance ratio of 10. Samples are learnt one-by-one and 'p' is set to 2000.

Hypothesis testing is used for detecting gradual drift. In this example, *Gmean* values obtained by the new classifier (added at the time step when change is detected) over subsequent time steps form the sequence for calculating $p(G_i|\mathscr{H}_0)$ and $p(G_i|\mathscr{H}_1)$. A likelihood ratio test using (14) is then performed with 'τ' set to 1.5. The test is performed at every alternate time step starting with 6 previous evaluations. If there is no change, the next test is performed with additional *Gmean* values from the next two evaluations i.e. a sequence of past 8 *Gmean* values. For this dataset, changes are detected mostly when sequence size is 10 or 12. Through likelihood ratio test, 15 batch WELMs are trained using ELM-Store over one complete rotation of spirals. Figs. 7 and 8 compare the performance of ESOS-ELM with those of WOS-ELM and DWM-CIL, respectively. The average *Gmean* values over entire dataset are reported in Table 1. The average *Gmean* values obtained by ESOS-ELM and DWM-CIL after the concepts re-occur are 80.8 and 80.3, respectively. WOS-ELM performs poorly on this gradually drifting dataset.

4.1.4. Electricity pricing domain (elect2)

The 'elect2' was first used by Harris [33] and has now become a real-world benchmark dataset for concept drift learning. This dataset was acquired from electricity supplier from New South Wales (NSW), Australia. It consists of 45,312 instances, each with 5 attributes. The first attribute is the day of week (1–7), followed by the period of day (1–48). The remaining three attributes represents NSW electricity demand, Victoria electricity demand, and the amount of electricity scheduled for transfer between the two states. The classification goal is to predict whether the NSW price will be higher than Victoria's price. The dataset contains

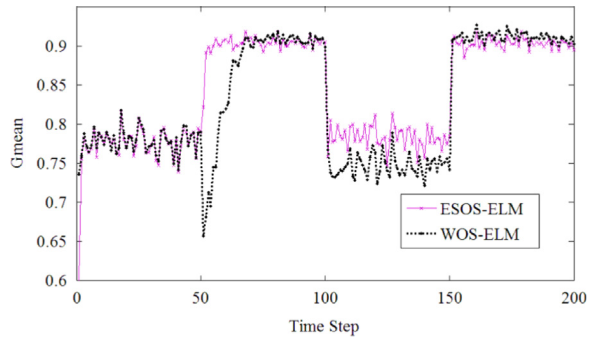

Fig. 5. ESOS-ELM and WOS-ELM comparison on SEA concepts.

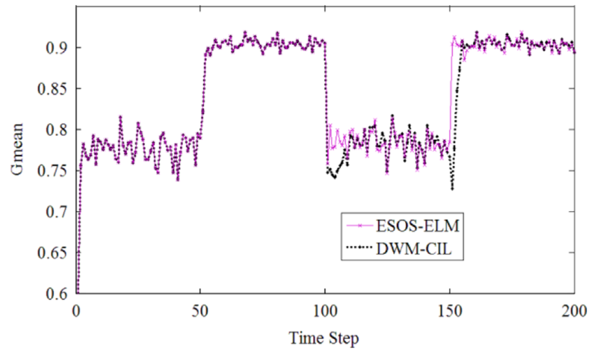

Fig. 6. ESOS-ELM and DWM-CIL comparison on SEA concepts.

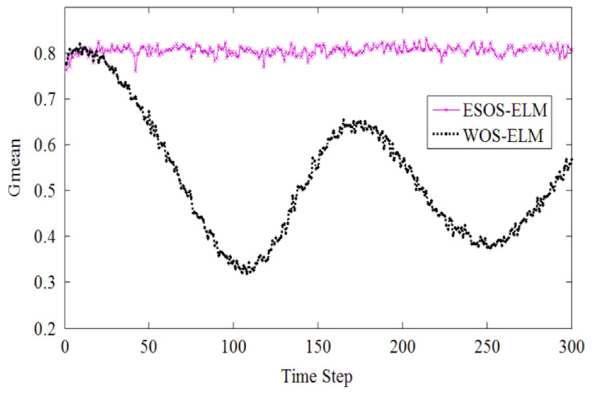

Fig. 7. ESOS-ELM and WOS-ELM comparison on rotating spiral.

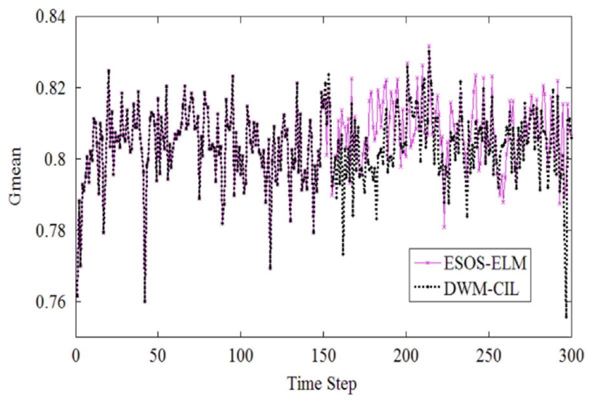

Fig. 8. ESOS-ELM and DWM-CIL comparison on rotating spiral.

natural concept drift. However, the original dataset has an imbalance ratio of less than 2 so we undersampled the minority class (class 1) such that imbalance ratio between the two classes is around 13.

Samples are presented one at a time. Instead of evaluating the learner's performance with each arriving sample, we evaluate whenever a minority class sample arrives, with 50 subsequent samples forming the validation set. Fig. 9 shows the performance comparison between ESOS-ELM and WOS-ELM. The average *Gmean* values over the entire dataset are reported in Table 1. ESOS-ELM and DWM-CIL obtained similar results. Concept drift and high imbalance make classification difficult for this dataset. *Gmean* values obtained using WOS-ELM fall to zero several times. The fluctuation in *Gmean* values obtained using ESOS-ELM is due to the fact that validation set contains very few minority class samples. Correct and wrong predictions of these samples obtain significantly different *Gmean* values. A bigger validation set can have many more minority class samples. However, it is a real-world online learning problem with high class imbalance in which minority class samples may not be available at each time step.

ESOS-ELM obtained significantly better performance than WOS-ELM on STAGGER concepts and Rotating Spiral datasets while the improvement is minor for the remaining two datasets. Clearly, WOS-ELM is not stable for non-stationary environments and need of an ELM related CIL method for non-stationary environments is justified. In recurring environments, ESOS-ELM responds faster to change in concepts than DWM-CIL.

For chunk-by-chunk case, ESOS-ELM is compared with a recently proposed Learn^{++}.NIE [16] method. Learn^{++}.NIE has shown superior generalization performance to earlier methods in recurring environments. Similar to ESOS-ELM, Learn^{++}.NIE does not store old minority class samples. For a fair comparison, we implemented Learn^{++}.NIE with ELM as the base classifier. In order to tackle the class imbalance problem, Learn^{++}.NIE trains a sub-ensemble of ELM for each data chunk. The number of classifiers in every sub-ensemble is set to 5. The numbers of hidden neurons for ELM are same as in ESOS-ELM except for Rotating Spiral dataset where 90 hidden neurons obtained the best average *Gmean*. The parameters a and b in Learn^{++}.NIE are set to 0.5 and 10, respectively [16]. Chunk sizes for STAGGER concepts, SEA concepts and rotating spiral are 40, 1000, and 2000, respectively. For elect2 dataset, chunk of 200 samples is used for training and the next 200 samples are used for testing. The same train and test procedure is continued for the remaining samples. At every time step, identical chunk of data is presented to both the methods in comparison. Learn^{++}.NIE exists for three different CIL measures i.e. *Gmean*, *F-measure* and *weighted recall measure*. We selected *Gmean* for determining voting weight of the sub-ensemble since ESOS-ELM also uses *Gmean* for its weight calculation. Moreover, other ELM related class imbalance learners such as WELM [5], OTER [6] and WOS-ELM [9] have been successfully evaluated with *Gmean* as the error measure. Table 2 shows *Gmean* values obtained by the two methods, along with the maximum number of hypotheses maintained by them. Figs. 10–13 show the performance comparison between ESOS-ELM and Learn^{++}.NIE.

For STAGGER concepts, Learn^{++}.NIE obtained slightly better *Gmean* than ESOS-ELM. For SEA concepts, superior performance of the new method is mainly due to the fact that it responds quickly to sudden drift, particularly after time step 50 (see Fig. 11). Finally, ESOS-ELM maintains higher *Gmean* on Rotating Spiral and elect2 dataset. These are concept drift learning problems with higher imbalance ratios than the previous two datasets. We observed that ESOS-ELM tackles the class imbalance problem more effectively compared to Learn^{++}.NIE(*gm*) for these highly imbalanced datasets. Moreover, it can be observed from Tables 1 and 2, ESOS-ELM obtains similar results for both chunk-by-chunk and one-by-one learning modes. The sensitivity of number of hidden neurons on the performance of ESOS-ELM for each dataset is shown in Fig. 14.

ESOS-ELM is primarily proposed for imbalanced datasets. It is of interest to see how the proposed method behaves on balanced datasets. Thus, we compare it with the original DWM method on datasets having relatively balanced classes. The majority classes in STAGGER and SEA datasets are undersampled first to make the class sizes equal while the original rotating spiral dataset already has balanced classes. All the three datasets have recurring concepts. The results are recorded in Table 3. ESOS-ELM obtains better performance than that of the original DWM even on balanced

Table 2
Classification results comparison for non-stationary environments (chunk-by-chunk).

Dataset	Algorithm	*Gmean*	Max no. of hypotheses
STAGGER	Learn^{++}.NIE(*gm*)	**98.6**	300
	ESOS-ELM	98.3	20
SEA	Learn^{++}.NIE(*gm*)	82.8	1000
	ESOS-ELM	**83.9**	18
Spiral	Learn^{++}.NIE(*gm*)	78.8	1500
	ESOS-ELM	**80.7**	44
Elect2	Learn^{++}.NIE(*gm*)	58.5	340
	ESOS-ELM	**61.1**	15

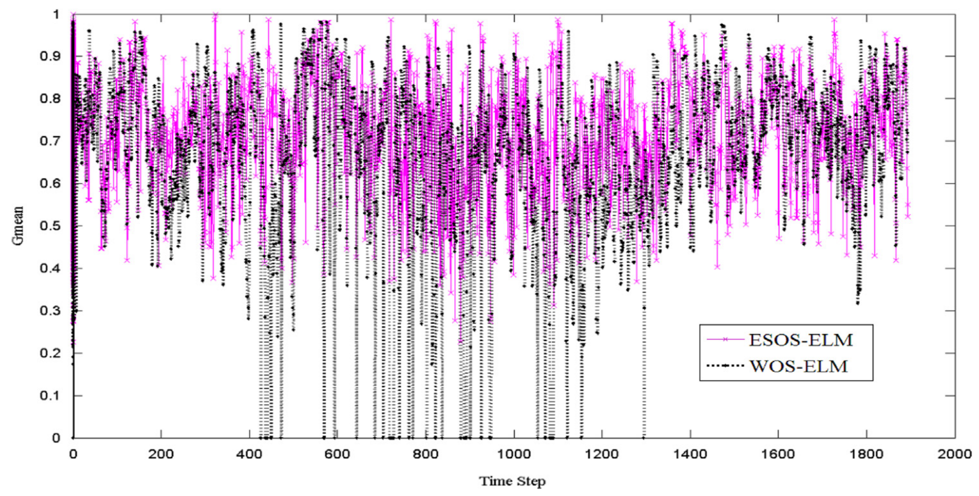

Fig. 9. ESOS-ELM and WOS-ELM comparison on the elect2 dataset.

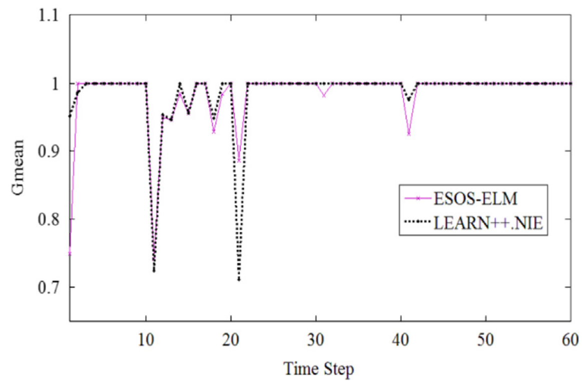

Fig. 10. ESOS-ELM and Learn^{++}.NIE(gm) comparison on STAGGER concepts.

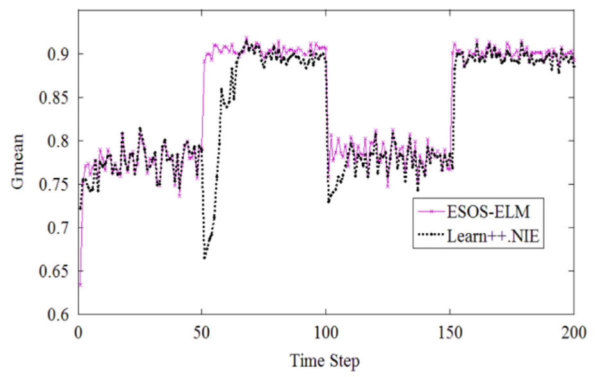

Fig. 11. ESOS-ELM and Learn^{++}.NIE(gm) comparison on SEA concepts.

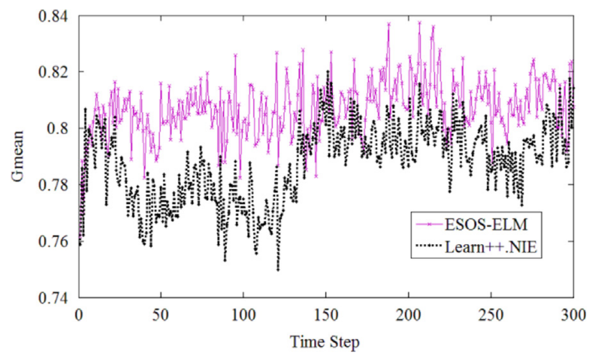

Fig. 12. ESOS-ELM and Learn^{++}.NIE(gm) comparison on rotating spiral.

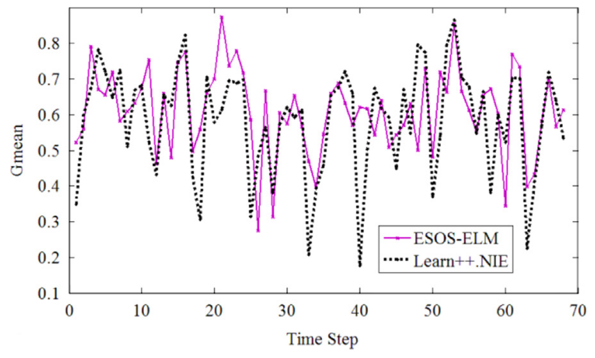

Fig. 13. ESOS-ELM and Learn^{++}.NIE(gm) comparison on elect2.

datasets. As explained earlier for the imbalanced case, ESOS-ELM responds faster to the concept drifts in the second half (recurring parts) than DWM which does not leverage prior knowledge of past concepts.

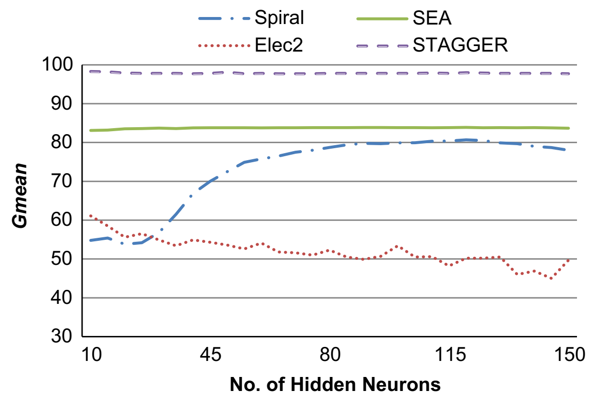

Fig. 14. Parameter tuning in ESOS-ELM.

Table 3
Classification results comparison for balanced datasets.

Dataset	Algorithm	*Gmean*
STAGGER (balanced)	DWM	94.1
	EOS-ELM-CIL	**94.8**
SEA (balanced)	DWM	83.4
	EOS-ELM-CIL	**83.7**
Sprial (balanced)	DWM	80.3
	EOS-ELM-CIL	**80.5**

4.2. Classification results on benchmark imbalanced datasets without concept drift

The performance of ESOS-ELM is also evaluated on the same 15 real-world imbalanced datasets that were previously used for performance evaluations of WOS-ELM, OTER and OS-ELM-SMOTE in [9]. 14 Datasets are taken from UCI repository [34] and the oil dataset is provided by Dr. Holte [35]. The details of these datasets are shown in Table 4. For a fair comparison, all the experimental settings i.e. validation set, testing set, chunk size and the number of hidden neurons are set identical to those of above CIL methods. Readers can refer to Ref. [9] for the number of hidden neurons, and other experimental details.

We first compare the performance of ESOS-ELM with those of WOS-ELM and OS-ELM-SMOTE in chunk-by-chunk mode. Same as for non-stationary datasets, we start with an ensemble size of 10. However, in some datasets the number of samples available for initialization of an individual classifier is very less than the number of hidden neurons. In such cases, a smaller ensemble size is selected. The smallest ensemble size used in this paper is 5. If sufficient samples are available for initialization, we recommend using ensemble size of 10. For sequential learning in stationary environments, evaluation is performed once at the end rather than at each time step [6,9,11,21]. Voting weights are updated using *Gmean* values obtained on the validation set. The results for chunk-by-chunk mode are recorded in Table 5. The new method obtained better or similar results to those of WOS-ELM and OS-ELM-SMOTE on all the datasets. We also compare performance of ESOS-ELM with those of WOS-ELM and OTER in one-by-one mode. The results for one-by-one learning are recorded in Table 6. As can be observed from the table, the new method has outperformed previous best methods in one-by-one learning. A comparison of *Gmean* values obtained by ESOS-ELM and WOS-ELM (previous best for one-by-one learning) is shown in Fig. 15. Note that the classification results for WOS-ELM, OS-ELM-SMOTE and OTER are taken from our previous work [9].

Friedman test [36] is used to confirm whether improvements achieved in Tables 5 and 6 are statistically significant than the

Table 4
Details of the datasets.

Dataset	# of Attributes	Imbalance ratio	# of Training samples	# of Testing samples	# of Validation samples
Page-block5	10	46.59	3973	1000	1000
Abalone15	8	39.55	3177	1000	500
Yeast5	8	28.09	1334	300	150
Car3	6	24.04	1528	300	200
Oil	49	21.85	837	200	100
Abalone9-18	8	16.4	631	100	100
Balance2	4	11.75	475	150	90
Satimage4	36	9.27	6135	600	600
mf-Morph10	6	9	1800	300	200
mf-Zernike10	47	9	1800	300	200
cmc2	9	3.42	1173	300	200
Transfusion	4	3.2	648	200	100
Haberman	3	2.77	366	60	40
Phoneme	5	2.4	4504	600	600
Pima	8	1.86	868	100	100

other online sequential algorithms in comparison. For both chunk-by-chunk and one-by-one learning, a post-hoc analysis is also applied since more than two CIL methods are compared.

For each dataset, the best performing algorithm is ranked as 1, the second best as 2, and so on. If classifiers produce same results, an average rank is assigned to each of them. For each classifier, mean rank over all the 15 datasets is calculated. Matlab function '*friedman*' is used for hypothesis testing. A null hypothesis that all the classifiers are equivalent and any differences among their average ranks are random is rejected, if the obtained p-value is lower than a specific confidence level α. For chunk-by-chunk case, the p-value obtained for all *Gmean* values is 3.372×10^{-6} thus a null hypothesis is rejected at $\alpha = 0.05$. In order to analyze differences among ESOS-ELM, WOS-ELM and OS-ELM-SMOTE, a post-hoc Nemenyi is adopted. The average ranks of ESOS-ELM, WOS-ELM and OS-ELM-SMOTE are 1, 2.2 and 2.8, respectively. If the rank difference between a pair of classifiers is larger than the critical difference (CD) at a certain confidence level, the two classifiers are considered statistically different. A CD of 0.855 is obtained at $p = 0.05$ [36]. The rank difference between ESOS-ELM and WOS-ELM is greater than CD so they are statistically different while the rank difference between WOS-ELM and OS-ELM-SMOTE is smaller than CD so they are statistically similar. Fig. 16 visualizes the Nemenyi post-hoc test at $p = 0.05$. The mean ranks over 15 datasets are shown in descending rank order on the x-axis. Statistically similar methods, WOS-ELM and OS-ELM-SMOTE in this case, are grouped with a horizontal bar [36].

For one-by-one case, the p-value obtained for all *Gmean* values is 2.4189×10^{-6} thus null hypothesis is rejected at $\alpha = 0.05$. The average ranks of ESOS-ELM, WOS-ELM and OTER are 1, 2.866 and 2.133, respectively. The rank difference between ESOS-ELM and WOS-ELM is found to be greater than CD so they are statistically different, while the rank difference between WOS-ELM and OTER is found to be smaller than CD so they are statistically similar. Fig. 17 visualizes the Nemenyi post-hoc test at $p = 0.05$. Statistically similar methods, WOS-ELM and OTER in this case, are grouped with horizontal bar.

As can be observed from Figs. 16 and 17, ESOS-ELM is statistically superior to the previous best methods in both chunk-by-chunk and one-by-one learning modes. In our previous work, WOS-ELM was found to be statistically superior to the other methods only in one-by-one but not in chunk-by-chunk [9].

Table 5
Classification results comparison for stationary environments (chunk-by-chunk).

Dataset	Chunk size	Algorithm	*Gmean*
Page-block5	100	WOS-ELM	91.7
		OS-ELM-SMOTE	91.6
		ESOS-ELM	**92.4**
Abalone15	100	WOS-ELM	70.9
		OS-ELM-SMOTE	70.8
		ESOS-ELM	**73.7**
Yeast5	75	WOS-ELM	80.2
		OS-ELM-SMOTE	78.6
		ESOS-ELM	**82.2**
Car3	50	WOS-ELM	95
		OS-ELM-SMOTE	94.1
		ESOS-ELM	**95.9**
Oil	50	WOS-ELM	79.5
		OS-ELM-SMOTE	80.4
		ESOS-ELM	**83.2**
Abalone9-18	50	WOS-ELM	84.3
		OS-ELM-SMOTE	77.1
		ESOS-ELM	**87.1**
Balance2	30	WOS-ELM	67.7
		OS-ELM-SMOTE	63.1
		ESOS-ELM	**69.6**
Satimage4	50	WOS-ELM	86.2
		OS-ELM-SMOTE	85.7
		ESOS-ELM	**86.3**
mf-Morph10	50	WOS-ELM	92.3
		OS-ELM-SMOTE	92.1
		ESOS-ELM	**92.5**
mf-Zernike10	50	WOS-ELM	87.5
		OS-ELM-SMOTE	88.3
		ESOS-ELM	**89**
cmc2	50	WOS-ELM	66
		OS-ELM-SMOTE	65.8
		ESOS-ELM	**66.1**
Transfusion	20	WOS-ELM	67.9
		OS-ELM-SMOTE	67.1
		ESOS-ELM	**68.3**
Haberman	20	WOS-ELM	63.6
		OS-ELM-SMOTE	63.5
		ESOS-ELM	**66**
Phoneme	50	WOS-ELM	84.2
		OS-ELM-SMOTE	83.8
		ESOS-ELM	**84.3**
Pima	50	WOS-ELM	75.6
		OS-ELM-SMOTE	76.5
		ESOS-ELM	**77.1**

4.3. Discussion on results and computational complexity

ESOS-ELM obtains better performance than the recently proposed Learn^{++}.NIE(*gm*) method on 3 out of 4 non-stationary datasets used in this paper. From the experiments, it is observed that ESOS-ELM tackles the class imbalance problem better than Learn^{++}. NIE(*gm*) and achieves comparable or slightly lower boast in performance in recurring environments, using fewer hypotheses. The new method uses similar approach as Dynamic weighted majority (DWM) voting method for tackling concept drifts.

Table 6
Classification results comparison for stationary environments (one-by-one).

Dataset	Algorithm	*Gmean*
Page-block5	WOS-ELM	91.7
	OTER	91.5
	ESOS-ELM	**92.4**
Abalone15	WOS-ELM	70.8
	OTER	69.9
	ESOS-ELM	**73.6**
Yeast5	WOS-ELM	80.2
	OTER	79.1
	ESOS-ELM	**82.4**
Car3	WOS-ELM	94.8
	OTER	93.1
	ESOS-ELM	**95.9**
Oil	WOS-ELM	79.6
	OTER	78.1
	ESOS-ELM	**83.4**
Abalone9-18	WOS-ELM	84.4
	OTER	82.8
	ESOS-ELM	**87.1**
Balance2	WOS-ELM	67.6
	OTER	66.8
	ESOS-ELM	**69.5**
Satimage4	WOS-ELM	85.7
	OTER	85.3
	ESOS-ELM	**86.3**
mf-Morph10	WOS-ELM	92.3
	OTER	92.1
	ESOS-ELM	**92.5**
mf-Zernike10	WOS-ELM	87.7
	OTER	86.5
	ESOS-ELM	**88.9**
cmc2	WOS-ELM	66
	OTER	66.2
	ESOS-ELM	**66.3**
Transfusion	WOS-ELM	67.9
	OTER	67.5
	ESOS-ELM	**68.2**
Haberman	WOS-ELM	63.7
	OTER	63.4
	ESOS-ELM	**66.2**
Phoneme	WOS-ELM	84.3
	OTER	84.2
	ESOS-ELM	**84.4**
Pima	WOS-ELM	75.6
	OTER	76
	ESOS-ELM	**77.2**

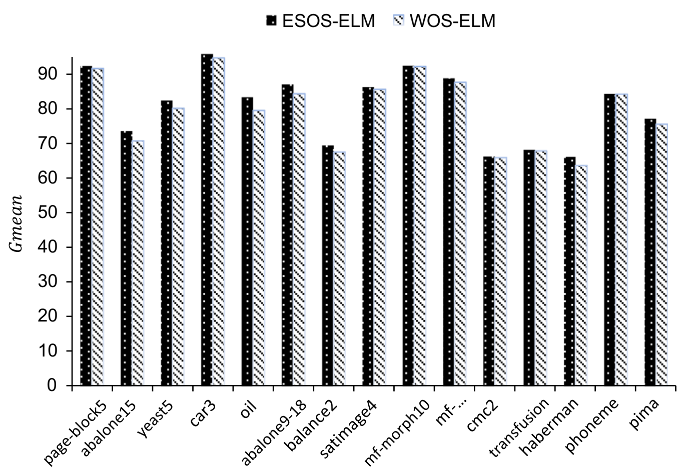

Fig. 15. *Gmean* comparison between ESOS-ELM and WOS-ELM.

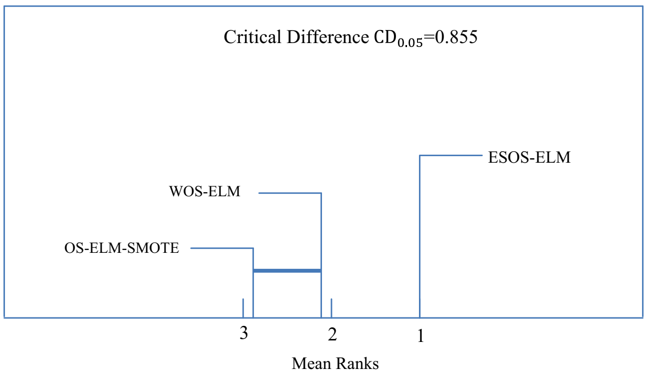

Fig. 16. Visualization of Nemenyi post-hoc test for chunk-chunk learning. The mean rank for each method over all the datasets is plotted on *x*-axis. Connected methods are not significantly different at $p=0.05$.

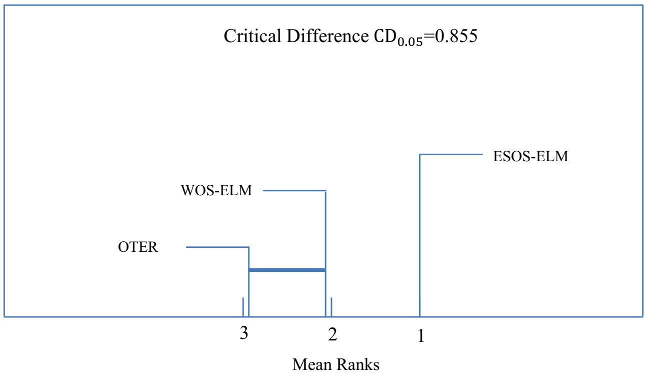

Fig. 17. Visualization of Nemenyi post-hoc test for one-one learning. The mean rank for each method over all the datasets is plotted on *x*-axis. Connected methods are not significantly different at $p=0.05$.

However, using ELM-Store we are able to achieve faster response to concept drift than DWM with recurring concepts. The new method obtains similar results in both one-by-one and chunk-by-chunk learning. Moreover, we have shown that ESOS-ELM is more effective than WOS-ELM for tackling the class imbalance problem, in both stationary and non-stationary environments.

For non-stationary datasets with recurring concepts, the new method shows improvement over DWM-CIL. At each time step, ELM-Store updates four matrices which is actually same (if not less) as single OS-ELM, in terms of computational complexity. Only 5 extra hypotheses (WELM) are stored for The STAGGER concepts, 3 for The SEA concepts and 29 extra hypotheses for The Rotating Spiral dataset. These numbers are much smaller than in Learn^{++}.NIE(*gm*) [16] which stores 5 hypotheses per chunk of data. The differences in numbers of stored hypotheses can be seen in Table 2. Thus, ESOS-ELM provides a computationally efficient alternative to Learn^{++}.NIE(*gm*) in recurring environments, with the added advantage of learning samples one-by-one. Similar to Learn^{++}.NIE(*gm*), our method does not store old minority class samples.

In this study we only used *Gmean* for determining voting weights in the ESOS-ELM method. We wish to explore ESOS-ELM with other CIL error criteria such as F-measure, area under ROC, weighted recall measure, etc. as in Learn++.NIE. Most of the recent Learn++ algorithms can add new classes in the middle of the learning process. We plan to add this capability to ESOS-ELM as well. Moreover, it will be interesting to investigate if changing number of hidden neurons with changing concepts has any significant performance gain. Although, we suspect that it would come with higher computational complexity.

5. Conclusions

In this paper, we have proposed an ensemble of subset online sequential extreme learning machine (ESOS-ELM) for class imbalance learning from drifting data streams. ESOS-ELM consists of a main ensemble for classification in the current imbalanced environment, an ELM-Store module for storing information of old concepts and a change detector for promptly detecting concept drifts. In ESOS-ELM, the main ensemble is trained with balanced subsets of the data stream. Similar to WOS-ELM, the new method works in both one-by-one and chunk-by-chunk learning modes. However, WOS-ELM only considers stationary environments while ESOS-ELM can tackle class imbalance problem in both stationary and non-stationary environments i.e. with or without concept drift. For recurring environments, the prior knowledge of old concepts is retained using the ELM-Store module. ELM-Store is a computationally efficient storage scheme derived from ELM theory. In non-stationary environments, the proposed framework obtained better or comparable performance to that of Learn^{++}.NIE(*gm*) using *Gmean* as the figure of merit. It is a computationally efficient alternative to Learn^{++}.NIE(*gm*) since it maintains significantly lower number of hypotheses. ESOS-ELM outperformed WOS-ELM, OTER and SMOTE on all 15 imbalanced datasets used in the previous work.

Acknowledgments

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

References

[1] H. He, E. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.
[2] M. Kubat, S. Matwin, Addressing the curse of imbalanced training sets: one-sided selection, in: Proceedings of the International Conference on Machine Learning, Nashville, TN, 1997, pp. 179–186.
[3] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[4] C.X. Ling, V.S. Sheng, Cost-sensitive learning and the class imbalance problem, Encycl. Mach. Learn. (2008) 231–235.
[5] W. Zong, G.-B. Huang, Y. Chen, Weighted extreme learning machine for imbalance learning, Neurocomputing 101 (2013) 229–242.
[6] Y. Kim, K.A. Toh, A.B.J. Teoh, H.L. Eng, W.Y. Yau, An online learning network for biometric scores fusion, Neurocomputing 102 (2013) 65–77.
[7] Y. Wang, Y. Zhang, Y. Wang, Mining data streams with skewed distribution by static classifier ensemble, Stud. Comput. Intell. 214 (2009) 65–71.
[8] H. Nguyen, E. Cooper, K. Kamei, Online learning from imbalanced data streams, in: Proceedings of the International Conference of Soft Computing and Pattern Recognition (SoCPaR), 2011, pp. 347–352.
[9] B. Mirza, Z. Lin, K.A. Toh, Weighted online sequential extreme learning machine for class imbalance learning, Neural Process. Lett. 38 (3) (2013) 465–486.
[10] S. Ozawa, S. Pang, N. Kasabov, Incremental learning of chunk data for online pattern classification, IEEE Trans. Neural Netw. 19 (6) (2008) 1061–1074.
[11] N.Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Trans. Neural Netw. 17 (6) (2006) 1411–1423.
[12] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Netw. 22 (10) (2011) 1517–1531.
[13] A. Shilton, M. Palaniswami, D. Ralph, A.C. Tsoi, Incremental training of support vector machines, IEEE Trans. Neural Netw. 16 (1) (2005) 114–131.
[14] T.R. Hoens, R. Polikar, N.V. Chawla, Learning from streaming data with concept drift and imbalance: an overview, Prog. Artif. Intell. 1 (1) (2012) 89–101.
[15] G. Ditzler, R. Polikar, N.V. Chawla, An incremental learning algorithm for non-stationary environments and class imbalance, in: Proceedings of the International Conference on Pattern Recognition, 2010, pp. 2997–3000.
[16] G. Ditzler, R. Polikar, Incremental learning of concept drift from streaming imbalanced data, IEEE Trans. Knowl. Data Eng. 25 (10) (2012) 2283–2301.
[17] J. Gao, W. Fan, J. Han, P.S. Yu, A general framework for mining concept-drifting streams with skewed distribution, in: Proceedings of the SIAM International Conference on Data Mining, 7, 2007.
[18] S. Chen, H. He, SERA: selectively recursive approach towards nonstationary imbalanced stream data mining, in: Proceedings of the International Joint Conference on Neural Networks, Atlanta, GA, 2009, pp. 522–529.
[19] S. Chen, H. He, Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach, Evol. Syst. 2 (1) (2011) 35–50.
[20] G.-B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.
[21] Y. Lan, Y.C. Soh, G.-B. Huang, Ensemble of online sequential extreme learning machine, Neurocomputing 72 (2009) 3391–3395.
[22] J. Cao, Z. Lin, G.-B. Huang, N. Liu, Voting based extreme learning machine, Inf. Sci. 185 (1) (2012) 66–77.
[23] J. Cao, Z. Lin, G.-B. Huang, Voting base online sequential extreme learning machine for multi-class classification, in: Proceedings of the IEEE ISCAS, Beijing, China, May, 2013.
[24] J.Z. Kolter, M.A. Maloof., Dynamic weighted majority: an ensemble method for drifting concepts, J. Mach. Learn. Res. 8 (2007) 2755–2790.
[25] K. Nishida, K. Yamauchi, T. Omori, ACE: adaptive classifiers-ensemble system for concept-drifting environments, Mult. Classif. Syst. (2005) 176–185.
[26] Y. Tang, Y. Zhang, N.V. Chawla, S. Krasser, SVMs modeling for highly imbalanced classification, IEEE Trans. Syst. Man Cybern.B Cybern. 39 (1) (2009) 281–288.
[27] J.R. Akbani, S. Kwek, N. Japkowicz, Applying support vector machines to imbalanced datasets, in: Proceedings of the 15th European Conference on Machine Learning, Pisa, Italy, 2004, pp. 39–50.
[28] R. Batuwita, V. Palade, FSVM-CIL: fuzzy support vector machines for class imbalance learning, IEEE Trans. Fuzzy Syst. 18 (3) (2010) 558–571.
[29] F. Chu, C. Zaniolo, Fast and light boosting for adaptive mining of data streams, Adv. Knowl. Discov. Data Min. LNCS 3056 (2004) 282–292.
[30] J.C. Schlimmer, R.H. Granger, Beyond incremental processing: tracking concept drift, in: Proceedings of the Fifth National Conference on Artificial Intelligence, 1, 1986, pp. 502–507.
[31] W.N. Street, Y. Kim, A streaming ensemble algorithm (SEA) for large-scale classification, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 377–382.
[32] G. Ditzler, R. Polikar, Benchmark datasets for evaluating concept drift/imbalanced data algorithms. Available from: ⟨http://users.rowan.edu/~polikar/research/NIE_data⟩, 2013. (last accessed 25.04.13).
[33] M. Harries, New South Wales, Splice-2 Comparative Evaluation: Electricity Pricing. Available from: ⟨ftp://ftp.cse.unsw.edu.au/pub/doc/papers/UNSW/9905.pdf.⟩, 1999(last accessed 25.04.13).
[34] A. Frank, A. Asuncion, UCI machine learning repository, University of California, Irvine, School of Information and Computer Sciences. Available from: ⟨http://archive.ics.uci.edu/ml⟩. (last accessed 25.04.13).
[35] M. Kubat, R. Holte, S. Matwin, Machine learning for the detection of oil spills in satellite radar images, Mach. Learn. 30 (1998) 195–215.
[36] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.

Bilal Mirza received the M.Sc. (signal processing) degree from Nanyang Technological University (NTU) in 2010. He is currently working towards the Ph.D. degree from NTU. His research interests include machine learning and its application in bio-signal processing, class imbalance and online sequential learning.

Zhiping Lin received the B.Eng. degree in control engineering from South China Institute of Technology, Canton, China in 1982 and the Ph.D. degree in information engineering from the University of Cambridge, England in 1987. He was with the University of Calgary, Canada for 1987–1988, with Shantou University, China for 1988–1993, and with DSO National Laboratories, Singapore for 1993–1999. Since February, 1999, he has been an Associate Professor at Nanyang Technological University (NTU), Singapore. He is also the Program Director of Bio-Signal Processing, Valens Centre of Excellence, NTU. Dr. Lin is currently serving as the Editor-in-Chief of Multidimensional Systems and Signal Processing after serving as an editorial board member for 1993–2004 and a CO-Editor for 2005–2010. He was an Associate Editor of Circuits, Systems and Signal Processing for 2000–2007 and an Associate Editor of IEEE Transactions on Circuits and Systems, Part II, for 2010–2011. He also serves as a reviewer for Mathematical Reviews. He is General Chair of the 9th International Conference on Information, Communications and Signal Processing (ICICS), 2013. His research interests include multidimensional systems and signal processing, statistical and biomedical signal processing, and more recently machine learning. He is the co-author of the 2007 Young Author Best Paper Award from the IEEE Signal Processing Society, Distinguished Lecturer of the IEEE Circuits and Systems Society for 2007–2008, and the Chair of the IEEE Circuits and Systems Singapore Chapter for 2007–2008.

Nan Liu received the B.Eng. degree in electrical engineering from University of Science and Technology Beijing, China, and the Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore. He is currently a Senior Research Scientist at the Department of Emergency Medicine, Singapore General Hospital. His research interests include pattern recognition, machine learning, and biomedical signal processing.