Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 8091267, 17 pages
http://dx.doi.org/10.1155/2016/8091267

Research Article

Adaptive Online Sequential ELM for Concept Drift Tackling

Arif Budiman, Mohamad Ivan Fanany, and Chan Basaruddin

Faculty of Computer Science, University of Indonesia, Depok, West Java 16424, Indonesia

Correspondence should be addressed to Arif Budiman; arif.budiman21@ui.ac.id

Received 29 January 2016; Accepted 17 May 2016

Academic Editor: Stefan Haufe

A machine learning method needs to adapt to over time changes in the environment. Such changes are known as concept drift. In this paper, we propose concept drift tackling method as an enhancement of Online Sequential Extreme Learning Machine (OS-ELM) and Constructive Enhancement OS-ELM (CEOS-ELM) by adding adaptive capability for classification and regression problem. The scheme is named as adaptive OS-ELM (AOS-ELM). It is a single classifier scheme that works well to handle real drift, virtual drift, and hybrid drift. The AOS-ELM also works well for sudden drift and recurrent context change type. The scheme is a simple unified method implemented in simple lines of code. We evaluated AOS-ELM on regression and classification problem by using concept drift public data set (SEA and STAGGER) and other public data sets such as MNIST, USPS, and IDS. Experiments show that our method gives higher kappa value compared to the multiclassifier ELM ensemble. Even though AOS-ELM in practice does not need hidden nodes increase, we address some issues related to the increasing of the hidden nodes such as error condition and rank values. We propose taking the rank of the pseudoinverse matrix as an indicator parameter to detect "underfitting" condition.

1. Introduction

Data stream mining is a data mining technique, in which the trained model is updated whenever new data arrive. However, the trained model must work in dynamic environments, where a vast amount of data not only is continuously generated but also keeps changing. This challenging issue is known as concept drift [1], in which the statistical properties of the input attributes and target classes shifted over time. Such shifts can make the trained model less accurate.

More methods for concept drift handling can be found in the literature [1], where the aim is to boost the generalization accuracy. These methods pursue an accurate, simple, fast, and flexible way to retain classification performance when the drift occurs. Ensemble classifier is a well-known way to retain the classification performance. The combined decision of many single classifiers (mainly using ensemble members diversification) is more accurate than single classifier [2]. However, it has higher complexity when handling multiple (consecutive) concept drifts.

One of the popular machine learning methods is Extreme Learning Machine (ELM) introduced by Huang et al. [3–7]. The ELM is a Single-Layer Feedforward Neural Network (SLFN) with fast learning speed and good generalization capability.

In this paper, we focused on the learning adaptation method as an enhancement to Online Sequential Extreme Learning Machine (OS-ELM) [8] and Constructive Enhancement OS-ELM (CEOS-ELM) [9]. We named it as adaptive OS-ELM (AOS-ELM). The AOS-ELM has capability to handle multiple concept drift problems, either changes in the number of attributes (virtual drift/VD) or the number of target classes (real drift/RD) or both at the same time (hybrid drift/HD), also for recurrent context (all concepts occur alternately) or sudden drift (new concept substitutes previous concepts) [10]. Our scope of attribute changes discussed in this paper is on the feature space concatenation widely used in data fusion, kernel fusion, and ensemble learning [11] and not on the feature selection (irrelevant features removal) methods [12]. We compared the performance with nonadaptive sequential ELM: OS-ELM and CEOS-ELM. We also compared the performance with ELM classifier ensembles as the common adaptive approach for concept drift solution. In the present study, although we focus on the adaptation aspect,

we address some possible change detection mechanisms that are suitable for our method.

A preliminary version of RD and its early results appeared in conference proceedings [14]. In this paper, we introduced the new scenarios in VD, HD, and consecutive drifts, either recurrent or sudden drift scenarios as well as theoretical background explanation. Our main contributions in this research area can be summarized as follows:

(1) We proposed simple adaptive method as enhancement to OS-ELM and CEOS-ELM for addressing concept drifts issue. Unlike ensemble systems [6, 13] that need to manage the complex combination of a vast number of classifiers, we pursue a single classifier for simple implementation while retaining comparable performance for handling multiple (consecutive) drifts.

(2) We introduced a simple unified platform to handle a hybrid drift (HD) when changes in the number of attributes and the number of target classes occurred at the same time.

(3) We elaborated how the AOS-ELM for transfer learning uses hybrid drift strategy. Transfer learning focuses on extracting the knowledge from one or more source task domains and applies the knowledge to a different target task domain [15]. Concept drift focuses on the time-varying domain with a small number of current data available. In contrast, transfer learning is not associated with time and requires the entire training and testing data set [16]. The example of transfer learning by using HD strategy is the transition from different data set sources but still related and with the same purpose. In this paper, we discussed the transfer learning on numeric handwritten MNIST [17] to alphanumeric handwritten USPS [18] recognition.

(4) Naturally, the AOS-ELM handling strategy was based on recurrent context. We devised an AOS-ELM strategy to handle sudden drift scenario by introducing output marginalization method. This method is also applicable for concept drift in a regression problem.

(5) We studied the effect of increasing the number of hidden nodes, which is treated as one of learning parameters, to improve the accuracy (other learning parameters are input weight, bias, activation function, and regularization factor). We proposed the evaluation parameter to predict the accuracy before the training was completed. We applied this assessment parameter actually to prevent "underfitting" or nonconvergence condition (the model does not fit the data well enough that makes accuracy performance dropped) when any learning parameter changes such as hidden nodes increased.

This paper is organized as follows. Section 2 explains some issues and challenges in concept drift, the background of ELM, and ELM in sequential learning. Section 3 presents the background theory and algorithm derivation of the proposed method. In Section 4, we focus on the empirical experiments to prove the methods and research questions in regression and classification problem. We use artificial and real data set. The artificial data sets are streaming ensemble algorithm (SEA) [19] and STAGGER [20], which are commonly used as benchmark in sequential learning. The real data sets are handwritten recognition data: MNIST for numeric [17] and USPS for alphanumeric classes [18]. We studied the effect of hidden nodes increase as one of the important learning parameters in Section 4.5. Section 7 discusses research challenges and future directions. The conclusion presents some highlights in Section 8.

2. Related Works

2.1. Notations. We specify the notations used throughout this article for easier understanding as follows:

(i) Matrix is written in uppercase bold (e.g., $\mathbf{X}$).

(ii) Vector is written in lowercase bold (e.g., $\mathbf{x}$).

(iii) The transpose of a matrix $\mathbf{X}$ is written as $\mathbf{X}^{\mathrm{T}}$. The pseudoinverse of a matrix $\mathbf{H}$ is written as $\mathbf{H}^{\dagger}$.

(iv) f, g will be used as nonlinear differentiable function (activation function), for example, sigmoid or tanh function.

(v) The amount of training data is N. Each input data $\mathbf{x}$ contains some d attributes. The target has m number of classes. An input matrix $\mathbf{X}$ can be denoted as $\mathbf{X}_{d\times N}$ and the target matrix $\mathbf{T}$ as $\mathbf{T}_{N\times m}$.

(vi) The hidden layer matrix is $\mathbf{H}$. The input weight matrix is $\mathbf{A}$. The output weight matrix is $\boldsymbol{\beta}$. The matrix $\Delta\mathbf{H}$ is the additional block portion of the matrix $\mathbf{H}$. The matrix $\mathbf{K}$ is the autocorrelation matrix of $\mathbf{H}^{\mathrm{T}}\mathbf{H}$. The inverse of matrix $\mathbf{K}$ is $\mathbf{P}$.

(vii) $\mathbf{H}$ can be denoted as $\mathbf{H}_{N\times L}$. $\mathbf{A}$ can be denoted as $\mathbf{A}_{d\times L}$ and $\boldsymbol{\beta}$ can be denoted as $\boldsymbol{\beta}_{L\times m}$. δL denotes the additional nodes number of L.

(viii) When the number of training data $N \to \infty$, we employed the online sequential learning method by updating model every time each new training pairs $(\mathbf{x}, \mathbf{t})$ are seen. $\mathbf{X}_{(0)}$ is the subset of input data at time $k = 0$ as the initialization stage. $\mathbf{X}_{(1)}, \mathbf{X}_{(2)}, \ldots, \mathbf{X}_{(k)}$ are the subset of input data at the next sequential time. Each subset may have different number of quantities. The corresponding label data is presented as $\mathbf{T} = [\mathbf{T}_{(0)}, \mathbf{T}_{(1)}, \mathbf{T}_{(2)}, \ldots, \mathbf{T}_{(k)}]$. We used the subscript font with parenthesis to show the sequence number.

(ix) We denote the training data from different S concepts (sources or contexts), using the symbol $\mathbf{X}_s$ for training data and $\mathbf{T}_s$ for target data. We used the subscript font without parenthesis to show the source number.

(x) We denote the drift event using the symbol $\overset{\ggg}{}_{\mathrm{VD}}$, where the subscript font shows the drift type. For example, Concept 1 has virtual drift event to be replaced by Concept 2 (sudden drift): $\mathbf{C}_1 \overset{\ggg}{}_{\mathrm{VD}} \mathbf{C}_2$. Concept 1 has real drift event to be replaced by Concept 1 and

Concept 2 recurrently (recurrent context) in the shuffled composition: $\mathbf{C}_1 \overset{\gg}{\underset{\text{RD}}{}} \text{shuffled}(\mathbf{C}_1, \mathbf{C}_2)$.

2.2. Concept Drift Strategies. In this section, we briefly explained the various concept drift solution strategies.

Gama et al. [1] explained that many concept drift methods have been developed, but the terminologies are not well established. According to Gama et al., the basic concept drift based on Bayesian decision theory in the classification problem for class output c and incoming data $\mathbf{X}$ is

$$P(c \mid \mathbf{X}) = P(c) \frac{P(\mathbf{X} \mid c)}{P(\mathbf{X})}. \quad (1)$$

Concept drift occurred when $P(c \mid \mathbf{X})$ has changed; for example, $\exists \mathbf{X} : P_{(0)}(\mathbf{X}, c) \neq P_{(1)}(\mathbf{X}, c)$, where $P_{(0)}$ and $P_{(1)}$ are, respectively, the joint distribution at times $t_{(0)}$ and $t_{(1)}$. Gama et al. categorized the concept drift types as follows:

(1) Real drift (RD) refers to changes in $P(c \mid \mathbf{X})$. The change in $P(c \mid \mathbf{X})$ may be caused by a change in the class boundary (the number of classes) or the class conditional probabilities (likelihood) $P(\mathbf{X} \mid c)$. The number of classes expanded and different class of data may come alternately, known as recurrent context. A drift, where new conditional probabilities replace the previous conditional probabilities while the number of classes remained the same, is known as sudden drift. Other terms are concept shift or conditional change [21].

(2) Virtual drift (VD) refers to the changes in the distribution of the incoming data (e.g., $P(\mathbf{X})$ changes). These changes may be due to incomplete or partial feature representation of the current data distribution. The trained model is built with additional data from the same environment without overlapping the true class boundaries. Other terms are feature change [21], temporary drift, or sampling shift.

Kuncheva [10, 22] explained the various configuration patterns of data sources over time as random noise, random trends (gradual changes), random substitutions (abrupt or sudden changes), and systematic trends (recurring context). The random noise will simply be filtered out. A gradual drift occurs when many concepts may reoccur alternately in the gradual stage for a certain period. A consecutive drift takes place when many previously active concepts might keep on changing alternately (recurring context) after some time. The sudden drift (abrupt changes or concept substitutions) is the type that at one time one concept is suddenly replaced by another concept.

Žliobaitė [13] proposed a taxonomy of concept drift tackling methods as shown in Figure 1. It describes the methods based on when the model is switched on (the "when" axis) and how the learners adapt to training set formation or design and parametrization of the base learner (the "how" axis). The "when" axis spans drift handling from trigger based to evolving based methods. The "how" axis spans drift handling from training set formation to model manipulation (or parametrization) methods.

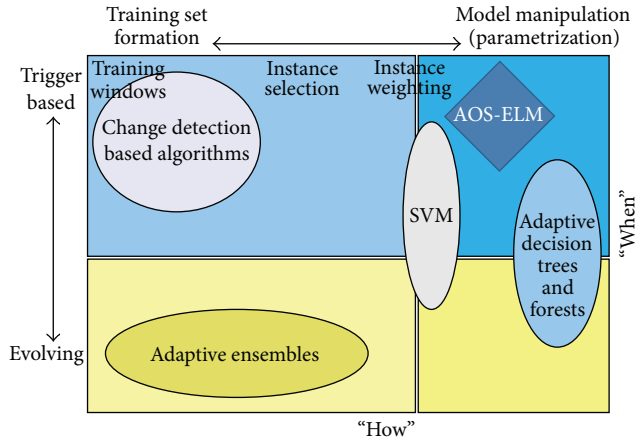


Figure 1: The taxonomy quadrant of adaptive supervised learning techniques. Popular concept drift handling methods are indicated by ellipses [13]. Our proposed method AOS-ELM is indicated by a dark blue diamond.

Žliobaitė [13] explained that most attention on the concept drift tackling methods is drawn to multiclassifier model selection and fusion rules, but little attention is drawn on the model construction of base classifier.

Gama et al. [1] proposed a complete online adaptive learning scheme that organized four modules: memory, change detection, learning, and loss estimation (see Figure 2). These modular components can be integrated, permuted, and combined with each other. The key modules are the learning and the change detection modules. Most methods focused on some subset or often mixtures of many types within certain concept drifts.

The learning module refers to the methods for the adaptation strategies of the predictive model. The learning module is categorized based on (i) how the model is updated when new data points are available (learning mode): retraining or incremental (online) modes; (ii) the behavior of predictive models on time-evolving data (model adaptation): a blind (evolving or implicit) based module or an informed (trigger or explicit) based module; (iii) the techniques for maintaining active predictive models (model management): a single model or ensemble model. The change detection module refers to drift detection. The change detection identifies change points or small time intervals when changes occur.

Each drift employed different solution strategies. The solution for RD is entirely different from VD. If the systematic changes are likely to reappear, we may want to keep past successful classifiers and simply reuse them. If the changes are gradual, we may use a moving window strategy on the training data. If the changes are abrupt, we can pause the existing static classifiers and then retrain the classifier using the new training data. Thus, it is hard to combine simultaneously many strategies at one time to solve many types of concept drift in just a simple platform.

2.3. ELM in Sequential Learning. In this section, we briefly explained the previous related works of ELM in sequential learning and adaptive environments.

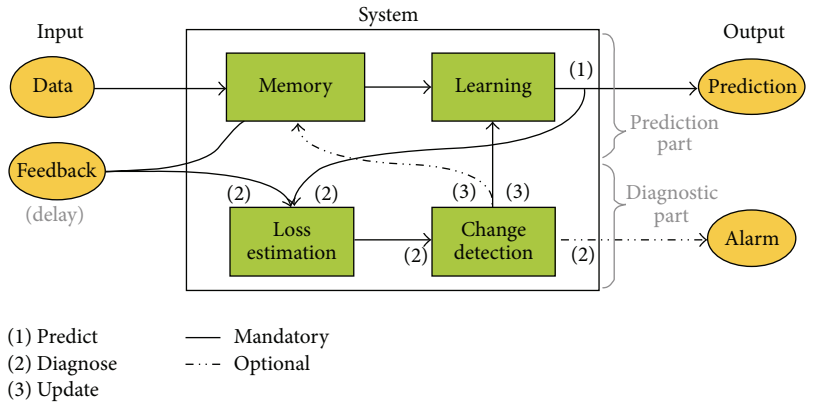


Figure 2: A generic scheme for an online adaptive learning algorithm from Gama et al. [1].

ELM is getting popularity thanks to its learning speed, generalization capability, and simplicity. Huang [5] explained the term "Extreme" meant to move beyond conventional artificial neural network learning that required iterative tuning. The ELM moves toward brain-like learning in which hidden neurons need not be tuned.

The output function of an SLFN with single hidden layer matrix $\mathbf{H}$ can be presented as the function of

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i \mathbf{H}(\mathbf{a}_i, \mathbf{b}_i, \mathbf{x}), \tag{2}$$

where $\mathbf{H} = g(\mathbf{AX} + \mathbf{b})$. All of these parameters defining the values of $\mathbf{H}$ elements are named as hidden node parameters [6].

The solution of ELM training with the smallest error can be obtained when the output weight $\boldsymbol{\beta}$ is approximated by

$$\widehat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{3}$$

where $\mathbf{H}^{\dagger}$ is the pseudoinverse of $\mathbf{H}$.

$\mathbf{H}^{\dagger}$ can be approximated by left pseudoinverse of $\mathbf{H}$ as

$$\widehat{\boldsymbol{\beta}} = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{T}. \tag{4}$$

We can use ridge regression or regularized least squares to be $\widehat{\boldsymbol{\beta}} = (\mathbf{H}^{\mathrm{T}}\mathbf{H} + \mathbf{I}/c)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{T}$.

Based on [4], Liang et al. [8] proposed online learning for ELM named OS-ELM. If we have $\widehat{\boldsymbol{\beta}}_{(0)}$ from $\mathbf{H}_{(0)}$ filled by the N_0 number of training data and N_1 incremental batch of data filled $\mathbf{H}_{(1)}$, the output weights $\widehat{\boldsymbol{\beta}}_{(1)}$ are approximated by

$$\widehat{\boldsymbol{\beta}}_{(1)} = \left(\begin{bmatrix}\mathbf{H}_{(0)}\\ \mathbf{H}_{(1)}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{H}_{(0)}\\ \mathbf{H}_{(1)}\end{bmatrix}\right)^{-1}\begin{bmatrix}\mathbf{H}_{(0)}\\ \mathbf{H}_{(1)}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{T}_{(0)}\\ \mathbf{T}_{(1)}\end{bmatrix}. \tag{5}$$

Both $\mathbf{H}_0$ and $\mathbf{H}_1$ have a different number of training data but have the same L number of hidden nodes.

If $\mathbf{K} = \mathbf{H}^{\mathrm{T}}\mathbf{H}$, then we can rewrite

$$\widehat{\boldsymbol{\beta}}_{(1)} = \mathbf{K}_{(1)}^{-1}\begin{bmatrix}\mathbf{H}_{(0)}\\ \mathbf{H}_{(1)}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{T}_{(0)}\\ \mathbf{T}_{(1)}\end{bmatrix}. \tag{6}$$

The OS-ELM assumes no changes in the number of hidden nodes. However, increasing the number of hidden nodes is required to improve the performance. A CEOS-ELM [9] has addressed this problem by adding hidden nodes in the sequential learning stage. So $\mathbf{H} = \begin{bmatrix}\mathbf{H}_{(0)} & \Delta\mathbf{H}_{(0)}\\ \mathbf{H}_{(1)} & \Delta\mathbf{H}_{(1)}\end{bmatrix}$. The submatrix $\Delta\mathbf{H}_{(0)}$ is set to a zero block matrix to simplify the computation in accordance with the fact that the previous data is not related to the new hidden nodes. The additional hidden nodes block matrix $\Delta\mathbf{H}_{(1)}$ for N_1 data has relation to the additional hidden nodes $\delta L_{(1)}$.

Then, we can rewrite $\mathbf{K}_{(1)}$ with $\Delta\mathbf{H}_{(1)}$ as

$$\widehat{\mathbf{K}}_{(1)} = \begin{bmatrix}\mathbf{H}_{(0)} & \mathbf{0}\\ \mathbf{H}_{(1)} & \Delta\mathbf{H}_{(1)}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{H}_{(0)} & \mathbf{0}\\ \mathbf{H}_{(1)} & \Delta\mathbf{H}_{(1)}\end{bmatrix}. \tag{7}$$

If $\widehat{\mathbf{P}} = \widehat{\mathbf{K}}^{-1}$ can be solved using block matrix inversion and Schur complement, then

$$\widehat{\boldsymbol{\beta}}_{(1)} = \widehat{\mathbf{P}}_{(1)}\begin{bmatrix}\mathbf{H}_{(0)} & \mathbf{0}\\ \mathbf{H}_{(1)} & \Delta\mathbf{H}_{(1)}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{T}_{(0)}\\ \mathbf{T}_{(1)}\end{bmatrix}. \tag{8}$$

It is important to note that both OS-ELM and CEOS-ELM did not address the concept drift issue; for example, when the number of attributes d in $\mathbf{X}_{d\times N}$ or the number of classes m in $\mathbf{T}_{N\times m}$ in data set has been added. In this paper, we categorized OS-ELM and CEOS-ELM as nonadaptive sequential ELM.

To the best of our knowledge, no previous single base ELM approach specifically addresses many concept drifts learning [6]. However, some papers [23, 24] already discussed how the ELM is implemented in adaptive environment.

van Schaik and Tapson [23] proposed Online Pseudoinverse Update Method (OPIUM). OPIUM is based on Greville's method as the incremental solutions to compute the pseudoinverse of matrix. The pseudoinverse computation can be solved incrementally as linear regression problems and can be adaptive which allows for nonstationary data. The derivation of OPIUM is equivalent to the OS-ELM if the condition $c_{(k)} \overset{\text{def}}{=} (\mathbf{I} - \mathbf{H}_{(k-1)}\mathbf{H}_{(k-1)}^{-1})\mathbf{H}_{(k)} = 0$ is met at each iteration. This condition implies that $\mathbf{H}_{(k)}$ is a linear

combination of the previous hidden layer $\mathbf{H}_{(k-1)}$ and the simpler derivation of (3) with right pseudoinverse becoming

$$\widehat{\boldsymbol{\beta}} = \mathbf{T}\mathbf{H}^{\dagger} = \mathbf{T}\mathbf{H}^{\mathrm{T}}\left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}. \quad (9)$$

van Schaik and Tapson defined $\boldsymbol{\psi}$ as the cross correlation matrix between $\mathbf{T}$ and $\mathbf{H}$ and $\boldsymbol{\theta}$ as the inverse of the autocorrelation $\mathbf{H}$, so $\boldsymbol{\beta} = \boldsymbol{\psi}\boldsymbol{\theta}$. According to Greville's method, the solution for $\boldsymbol{\psi}_{(k)} = \boldsymbol{\psi}_{(k-1)} + \mathbf{T}_{(k)}\mathbf{H}_{(k)}^{\mathrm{T}}$. And the solution for $\boldsymbol{\theta}_{(k)} = (\mathbf{H}_{(k-1)}\mathbf{H}_{(k-1)}^{\mathrm{T}} + \mathbf{H}_{(k)}\mathbf{H}_{(k)}^{\mathrm{T}})^{-1}$, or in short writing $\boldsymbol{\theta}_{(k)} = f(\boldsymbol{\theta}_{(k-1)}, \mathbf{H}_{(k)})$.

van Schaik and Tapson proposed a simplified version named OPIUM light by computing only the on-diagonal element of $\boldsymbol{\theta}_{(k)}$. van Schaik and Tapson applied the OPIUM light for nonstationary data by using different weight α in determining $\boldsymbol{\beta}_{(k)}$ for the most recent pair $(\mathbf{T}_{(k)}, \mathbf{H}_{(k)})$ appropriate for nonstationary mapping, which are $\boldsymbol{\psi}_{(k)} = (2-\alpha)\boldsymbol{\psi}_{(k-1)} + \alpha\mathbf{T}_{(k)}\mathbf{H}_{(k)}^{\mathrm{T}}$ and $\boldsymbol{\theta}_{(k)} = ((2-\alpha)\mathbf{H}_{(k-1)}\mathbf{H}_{(k-1)}^{\mathrm{T}} + \alpha\mathbf{H}_{(k)}\mathbf{H}_{(k)}^{\mathrm{T}})^{-1}$.

In our opinion, OPIUM only tackled the real drift case with discriminant function boundary shift in the streaming data (e.g., the frequency shift of sine wave). They implemented the weighting α as a nonstationary mapping parameter between input and output vectors.

Cao et al. [24] proposed two-phase classification algorithm: first, weighted ensemble classifier based on ELM (WEC-ELM) algorithm, which can dynamically adjust classifier and the weight of training uncertain data to solve the problem of concept drift, and second, an uncertainty classifier based on ELM (UC-ELM) algorithm designed for the classification of unknown data streams, which considers attribute (tuple) value and its uncertainty, thus improving the efficiency and accuracy. When concept drift occurs, WEC-ELM will dynamically adjust the classifiers and the weight of training data, thus a new classifier will be added to the ensemble until it reached a preset maximum and then removed the worst-performing classifier. UC-ELM is designed for the classification of uncertain data streams, which has attributes (tuples) and its uncertainty values. The UC-ELM evaluated uncertainty value for every newly arrived attribute and decided based on the probability of the new attributes belonging to each class, thus improving the efficiency and accuracy. In our opinion, WEC-ELM is categorized as evolving based method by selecting the best-performing classifier, and UC-ELM addressed virtual drift problem by using uncertainty attributes selection.

Most ELM work in adaptive environments addressed for particular drift case which may be impractical for other cases. We pursue a simple unified platform that has the capability to handle many (consecutive) drift cases.

3. Proposed Method

3.1. Theoretical Background of AOS-ELM. In sequential learning, some partial training data arrives in time sequential fashion: $\{(\mathbf{x}_{(0)}, \mathbf{t}_{(0)}), (\mathbf{x}_{(1)}, \mathbf{t}_{(1)}), \ldots, (\mathbf{x}_{(\mathrm{k})}, \mathbf{t}_{(k)})\}$. Learning is the process of constructing function $\widehat{\boldsymbol{\beta}}$ to map between observation and its nature called (class) [25]. When the number of training data $N \to \infty$, we need to address the expected value of $\boldsymbol{\beta}_{(\infty)} = \widehat{\boldsymbol{\beta}}$.

Learning from the data $\mathbf{D}_n$ is the process to select a function $\boldsymbol{\beta}_n$ from a class of $\mathfrak{B}$ by minimizing the empirical squared error $e_n(\boldsymbol{\beta}) = (1/n)\sum_{i=1}^{n}(\mathbf{H}_i\boldsymbol{\beta} - \mathbf{T}_i)^2$ with the error probability $L(\boldsymbol{\beta}_n) = P\{\mathbf{I}_{\{\mathbf{H}\boldsymbol{\beta}_n\}} \neq \mathbf{T} \mid \mathbf{D}_n\}$ of the resulting classifier. According to [25], the empirical squared error minimization is consistent under general conditions.

Theorem 1. *Assume that $\mathfrak{B}$ is a totally bounded class of functions. If $\boldsymbol{\beta}_n \in \mathfrak{B}$, then the classification rule obtained by minimizing the empirical squared error over $\mathfrak{B}$ is strongly consistent; that is,*

$$P\left\{\lim_{n\to\infty} L\left(\boldsymbol{\beta}_{\mathbf{n}}\right) = L^{*}\right\} \longrightarrow 1. \quad (10)$$

Based on Law of Large Numbers (LLN) theorem [26] and Theorem 1, in sequential learning with the number of training data $N \to \infty$, we can make sure that the consistency of expected value of learning model is $\widehat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}$.

The concept drift refers to an online supervised learning model when the relation between the input data and the target variable changes over time [1]. If the learning model from Concept 1 $\widehat{\boldsymbol{\beta}_1} \in \mathfrak{B}_1$ is bounded by hypothesis space $\mathbb{R}^{m_1}$ and feature space $\mathbb{R}^{d_1}$ and the learning model from Concept 2 $\widehat{\boldsymbol{\beta}_2} \in \mathfrak{B}_2$ is bounded by hypothesis space $\mathbb{R}^{m_2}$ and feature space $\mathbb{R}^{d_2}$, we defined the real drift as when the hypothesis space $\mathbb{R}^{m_1}$ has changed to $\mathbb{R}^{m_2}$. We scoped the definition for $m_2 > m_1$ dimension changes. The virtual drift is when the feature space $\mathbb{R}^{d_1}$ has changed to $\mathbb{R}^{d_2}$. We scoped the definition for $d_2 > d_1$ dimension changes.

To achieve the consistency of minimized square error in the new hypothesis space or new feature space, the learning model needs a transition map from the former space to the new space. The learning model $\widehat{\boldsymbol{\beta}_1}$ needs a transition space before it converges to the new learning model $\widehat{\boldsymbol{\beta}_2} \in \mathfrak{B}_2 \subset \mathbb{R}^{m_2}$. Our transition space idea was inspired by geometric approach for solving many problems in the fields of pattern recognition and machine learning [27, 28].

For transition space, we propose two approaches: (i) assign the random coordinates in the new concept space and (ii) assign the equivalent projection coordinates in the new design space. The first approach is suitable for VD scenario, in which we assigned the new random coordinates as the new input weight parameters. The second approach is suitable for RD situation, by setting the equivalent projection coordinates in the new space (e.g., (X_1) in 1D coordinate has corresponding 2D projection coordinates as $(X_1, 0)$).

Here, we relate the ELM theory to the context of AOS-ELM concept drift scenarios (see Table 1) as follows.

Scenario 1 (virtual drift (VD)). Huang et al. [6] explained interpolation theory from ELM point of view as stated by the following description.

Theorem 2. *Given any small positive value $\epsilon > 0$, any activation function which is infinitely differentiable in any interval, and N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbf{R}^d \times \mathbb{R}^m$,*

Table 1: Concept drift scenarios, compared methods, and sequential patterns.

(a) The experiment design scenarios

Data set	Virtual drift	Real drift	Hybrid drift	Compared methods
SEA	—	√	—	OS-ELM, CEOS-ELM, Kolter [20]
STAGGER	—	√	—	OS-ELM, CEOS-ELM, Kolter [20]
MNIST	√	√	√	OS-ELM, Offline ELM, ELM ensemble
MNIST + USPS	—	√	√	OS-ELM, offline ELM, ELM ensemble

(b) Concept drift sequential patterns

Data set	Sequential patterns scenarios	Cause of shift
SEA	Sudden change	Linear discriminant function
STAGGER	Sudden change	Logical discriminant rule
MNIST	Sudden change and recurring context	Additional attributes or classes
USPS	Recurring context	Additional attributes or classes

there exists $L < N$ such that, for any input weight and bias pair $\{\mathbf{a}_i, \mathbf{b}_i\}_{i=1}^{L}$ randomly generated from any interval of $\mathbf{R}^d \times \mathbb{R}$, according to any continuous probability distribution, with probability one, $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\| < \epsilon$. Furthermore, if $L = N$, then with probability one, $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\| = 0$.

According to Theorem 2 and Learning Principle I of ELM Theory [5], the input weight and bias as hidden nodes $\mathbf{H}$ parameters are independent of training samples and their learning environment through randomization. Their independence is not only in initial training but also in any sequential training stages. Thus, we can adjust the input weight and bias pair $\{\mathbf{a}_i, \mathbf{b}_i\}_{i=1}^{L}$ on any sequential stages and still make sure with probability one that $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\| < \epsilon$.

Scenario 2 (real drift (RD)). Huang et al. [6] explained universal approximation capability of ELM as described by the following theorem.

Theorem 3. *Given any nonconstant piecewise continuous function $g : \mathbb{R}_d \to \mathbb{R}$, if span $\{g(\mathbf{a}, \mathbf{b}, \mathbf{x}) : (\mathbf{a}, \mathbf{b})\mathbb{R}_d \times \mathbb{R}\}$ is dense in L2, for any continuous target function f and any function sequence $\{g(\mathbf{a}_i, \mathbf{b}_i, \mathbf{x})\}_{i=1}^{L}$ randomly generated according to any continuous sampling distribution, $\lim_{L\to\infty}\|f - f_L\| = 0$ holds with probability one if the output weights $\boldsymbol{\beta}_i$ are determined by ordinary least square to minimize $\|f(\mathbf{x}) - \Sigma_{L\,i=1}\boldsymbol{\beta}_i g(\mathbf{a}_i, \mathbf{b}_i, \mathbf{x})\|$.*

Based on Theorem 3 and inspired by the related works [9, 14], we devised the AOS-ELM real drift capability by modifying the output matrix with zero block matrix concatenation to change the size dimension of the matrix without changing the value. Zero block matrix has meant the previous $\boldsymbol{\beta}_{(k-1)}$ has no knowledge about the new concept. ELM can approximate any complex decision boundary, as long as the output weights $\boldsymbol{\beta}_i$ are determined by ordinary least square to keep the minimum.

3.2. AOS-ELM Algorithms. In this section, we presented the AOS-ELM pseudocodes (the Matlab source code, data set, and demo file implementation are available at https://github.com/abudiman250172/adaptive-OS-ELM) in the kth sequential with $\mathbf{X}_{(k)}$ training input and $\mathbf{T}_{(k)}$ target to update $Model_{(k)}$.

Basically, we have three pseudocodes, namely, OSELM-Seq (Algorithm 1) as OS-ELM and CEOS-ELM pseudocodes; AOSELMVDSeq (Algorithm 2) as AOS-ELM pseudocodes tackling virtual drift; and AOSELMRDSeq (Algorithm 3) as AOS-ELM pseudocodes for addressing real drift. We can combine the pseudocodes together to form a hybrid drift Algorithm 4. We can increase the hidden nodes using CEOS-ELM in Algorithm 1 after AOSELMVDSeq or AOSELMRD-Seq. For initialization, basically we can use any ordinary ELM initialization in offline learning mode.

For sudden drift scenario, we proposed output marginalization method by adding the new output nodes when the new concept presented (see Figure 3) and marginalized the output result by defining that $\mathbf{Y}_s$ class of concept S is $=\arg\max_{y_s}\mathbf{T}(y_s)$. We scoped that the new concept has the same output nodes quantity with the previous concept. Output marginalization is by shifting the ELM output to the output nodes belonging to the new concept and ignoring the previous concept output nodes. This strategy is similar with classifier pruning in ELM ensemble. However, in output marginalization, we can reactivate the previous concepts by shifting back to the previous output nodes. If we want to forget the last concept totally, we can quickly delete the previous output nodes without impacting the generalization performance, or we can increase the hidden nodes at the same time with the drift event.

In regression, because we have only one output node, then we can employ sudden drift scenario by amplifying the related output node of the concept with a constant value that makes the maximum output $\mathbf{Y}_s$ approximated to 1.

The systematic rules make AOS-ELM more flexibe to handle complex consecutive drifts scenario. The AOS-ELM only stored the previous output weight $\boldsymbol{\beta}_{L\times m}$ and autocorrelation $\mathbf{K}_{L\times L}$. The autocorrelation $\mathbf{K}$ did not keep the training data. This makes AOS-ELM scalable for big streaming data without impacting the computation performance.

To improve the accuracy, we define the target values $\in \{0, 1\}$, so that $\mathbf{Y}$ class is $=\arg\max_{y}\mathbf{T}(y)$. According to [29], the target values $\in \{0, 1\}$ are equivalent with $\in \{-1, 1\}$.

Require: $\mathbf{X}_{(k)} \in [-1, 1]\mathbb{R}^{d\times N}$, $\mathbf{T}_{(k)} \in [0, 1]\mathbb{R}^{N\times m}$, $\mathbf{A}_{(k)}$, $\mathbf{b}_L$, $\mathbf{K}_{(k-1)}$, $\boldsymbol{\beta}_{(k-1)}$
Ensure: $\boldsymbol{\beta}_{(k)}$, $\mathbf{K}_{(k)}$
(1) Compute $\mathbf{H}_{(k)} = g(\mathbf{A}_{(k)} \cdot \mathbf{X}_{(k)} + \mathbf{b}_L)$
(2) **if** *IncreaseHiddenNodes* == **true then**
(3) $\Delta\mathbf{A}_{d\times\delta L} = RandomNumbers([-1, 1], \mathbb{R}^{d\times\delta L})$
(4) $\Delta\mathbf{b}_{\delta L} = RandomNumbers([-1, 1], \mathbb{R}^{\delta L})$
(5) $\mathbf{A}_{(k)} = [\mathbf{A}_{(k)}\Delta\mathbf{A}_{d\times\delta L}]$
(6) $\mathbf{b}_L = [\mathbf{b}_{(L)}\Delta\mathbf{b}_{\delta L}]$
(7) Compute $\Delta\mathbf{H}_{(k)} = g(\Delta\mathbf{A}_{d\times\delta L} \cdot \mathbf{X}_{(k)} + \Delta\mathbf{b}_{\delta L})$
(8) Compute $\mathbf{K}_{(k)} = f(\mathbf{K}_{(k-1)}, \mathbf{H}_{(k)}, \Delta\mathbf{H}_{(k)})$
(9) Compute $\boldsymbol{\beta}_{(k)} = f(\boldsymbol{\beta}_{(k-1)}, \mathbf{K}_{(k)}, \mathbf{H}_{(k)}, \Delta\mathbf{H}_{(k)}, \mathbf{T}_{(k)})$ {Using CEOS-ELM Method}
(10) **else**
(11) Compute $\mathbf{K}_{(k)} = \mathbf{K}_{(k-1)} + \mathbf{H}_{(k)}^{\mathrm{T}}\mathbf{H}_{(k)}$
(12) Compute $\boldsymbol{\beta}_{(k)} = f(\boldsymbol{\beta}_{(k-1)}, \mathbf{K}_{(k)}, \mathbf{H}_{(k)}, \mathbf{T}_{(k)})$
(13) **end if**
(14) **return** $\boldsymbol{\beta}_{(k)}$, $\mathbf{K}_{(k)}$

Algorithm 1: Algorithm OSELMSeq {OS-ELM sequential}.

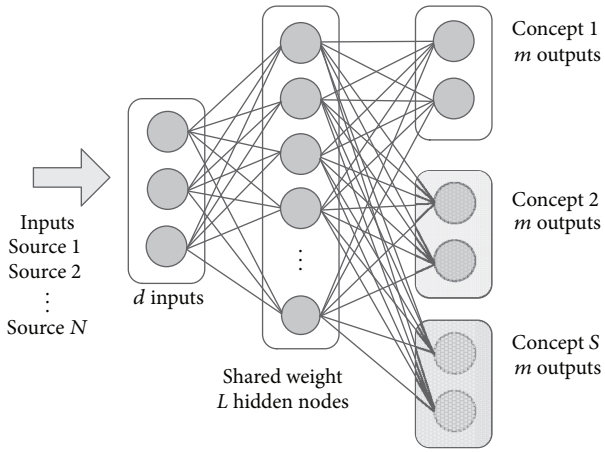


Figure 3: Output marginalization in AOS-ELM. The new block of output nodes assembled when the new concept S presented. Each concept has the same m output nodes quantity. Total output nodes become $S \times m$ output nodes.

4. Experiments

4.1. Experiments Design in Classification. To verify our method, we designed some experiments with the following purposes:

(i) To investigate the effectiveness of AOS-ELM on tackling three concept drift scenarios (VD, RD, and HD) in two sequential patterns (sudden changes and recurring context). We used various data set starting with synthetic data set (SEA, STAGGER) and then with real data set in handwritten recognition (MNIST, USPS). Each data set has different drift characteristics. This experiment is presented in Sections 4.2 and 4.4. We also demonstrated the AOS-ELM capability as drift detection role in Section 4.3 using SEA data set.

(ii) To investigate the effectiveness of AOS-ELM on transfer learning to combine different data set sources. This experiment is presented in Section 4.4 using two data set sources (MNIST and USPS) in handwritten recognition problem.

(iii) To investigate the effect of hidden nodes increase in the drift events and how it impacts performance. This experiment is presented in Section 4.5.

We used Matlab™ running on Microsoft Windows™ Computer with 4-core 2.5 GHz processor and 8 GB memory. Our experiments are organized as follows:

(1) Simulation benchmark tests on the data sets commonly used in concept drift handling of stream data, for example, SEA [19] and STAGGER [20] (see Table 2(a)). Both data sets are binary classification problem. SEA has 3 inputs with random integer values from 0 to 9. STAGGER has three inputs with multiple category values from 1 to 3 (total inputs are 9). SEA and STAGGER are the examples of concept drift caused by discriminant function changes while the number of attributes and classes from all concepts is still the same. The change type is sudden drift. The expected result is that the classifier has good performance for the newest concept [22].

(2) We tested our algorithm with real-world public data sets from MNIST numeric (0 to 9) [17] and the USPS alphanumeric (A to Z, 0 to 9) handwritten data set [18]. We used original grey-level image attributes $[X_{\text{grey}}]$ of MNIST data set and the combination of $[X_{\text{grey}}]$ with additional attributes from the 9×9 bins histogram of orientated gradients (X_{HOG}) of grey-level image features [30]. For USPS, we added more data with Gaussian random and salt-pepper noises. Refer to Table 2(a) for detailed data set information.

(3) We designed the initial input weights and bias based on robust OS-ELM with regularization scalar c (ROS) [31] and then based on initial random from the normal distribution (NORM). The activation function is sigmoid. The pseudoinverse function is the orthogonal projection using ridge regularization.

(4) Let us define the following concept as

(i) $\mathbf{C}_1$ is MNIST$[X_{\text{grey}}]$ class (1–6),
(ii) $\mathbf{C}_2$ is MNIST$[X_{\text{grey}}]$ class (7–10),
(iii) $\mathbf{C}_3$ is MNIST$[X_{\text{grey}}X_{\text{HOG}}]$ class (1–6),
(iv) $\mathbf{C}_4$ is MNIST$[X_{\text{grey}}X_{\text{HOG}}]$ class (7–10),
(v) $\mathbf{C}_5$ is MNIST$[X_{\text{grey}}X_{\text{HOG}}]$ class (1–10),
(vi) $\mathbf{C}_6$ is USPS$[X_{\text{grey}}X_{\text{HOG}}]$ class (1–10, A–Z).

We followed the simulated concept drift methods in Dries and Rückert [32]. We simulated sudden drift by splitting the composition into two groups, for example, $\mathbf{C}_1$ and $\mathbf{C}_2$, and recurring context by

Require: $\mathbf{X}_{(k)} \in [-1, 1]\mathbb{R}^{d\times N}$
$\mathbf{A}_{(k-1)} = Model_{(k-1)} \cdot \mathbf{A}$
Ensure: $\mathbf{A}_{(k)}$
(1) $d_{(k-1)} = SizeofAttributes(Model_{(k-1)})$
(2) $d_{(k)} = SizeofAttributes(\mathbf{X}_{(k)})$
(3) **if** $d_{(k)} > d_{(k-1)}$ **then**
(4) $\delta d = d_{(k)} - d_{(k-1)}$
(5) $\Delta\mathbf{A}_{\delta d\times L} = RandomNumbers([-1, 1], \mathbb{R}^{\delta d\times L})$
(6) $\mathbf{A}_{(k)} = \begin{bmatrix} \mathbf{A}_{(k-1)} \\ \Delta\mathbf{A}_{\delta d\times L} \end{bmatrix}$
{The construction of input weight in virtual drift scenario}
(7) **else**
(8) $\mathbf{A}_{(k)} = \mathbf{A}_{(k-1)}$
(9) **end if**
(10) **return** $\mathbf{A}_{(k)}$

Algorithm 2: Algorithm AOSELMVDSeq {AOS-ELM sequential-virtual drift}.

Require: $\mathbf{T}_{(k)} \in [0, 1]\mathbb{R}^{N\times m}$
$\boldsymbol{\beta}_{(k-1)} = Model_{(k-1)} \cdot \boldsymbol{\beta}$
Ensure: $\boldsymbol{\beta}_{(k-1)}$
(1) $m_{(k-1)} = SizeofClasses(Model_{(k-1)})$
(2) $m_{(k)} = SizeofClasses(\mathbf{T}_{(k)})$
(3) **if** $m_{(k)} > m_{(k-1)}$ **then**
(4) $\delta m = m_{(k)} - m_{(k-1)}$
(5) $\Delta\boldsymbol{\beta}_{L\times\delta m} = \mathbf{0}$
(6) $\boldsymbol{\beta}_{(k-1)} = [\boldsymbol{\beta}_{(k-1)}\Delta\boldsymbol{\beta}]$
{The construction of output weight in real drift scenario}
(7) **end if**
(8) **return** $\boldsymbol{\beta}_{(k-1)}$

Algorithm 3: Algorithm AOSELMRDSeq {AOS-ELM sequential-real drift}.

Require: $\mathbf{X}_{(k)} \in [-1, 1]\mathbb{R}^{d\times N}$, $\mathbf{T}_{(k)} \in [0, 1]\mathbb{R}^{N\times m}$,
$\mathbf{A}_{(k-1)} = Model_{(k-1)} \cdot \mathbf{A}$
$\mathbf{b}_L = Model_{(k-1)} \cdot \mathbf{b}$
$\mathbf{K}_{(k-1)} = Model_{(k-1)} \cdot \mathbf{K}$
$\boldsymbol{\beta}_{(k-1)} = Model_{(k-1)} \cdot \boldsymbol{\beta}$
Ensure: $Model_{(k)}$
(1) $\boldsymbol{\beta}_{(k-1)} = AOSELMRDSeq(\mathbf{T}_{(k)}, Model_{(k-1)})$
{The construction of output weight in real drift scenario}
(2) $\mathbf{A}_{(k)} = AOSELMVDSeq(\mathbf{X}_{(k)}, Model_{(k-1)})$
{The construction of input weight in virtual drift scenario}
(3) $(\boldsymbol{\beta}_{(k)}, \mathbf{K}_{(k)}) = OSELMSeq(\mathbf{X}_{(k)}, \mathbf{T}_{(k)}, \mathbf{A}_{(k)}, \mathbf{b}_L, \mathbf{K}_{(k-1)}, \boldsymbol{\beta}_{(k-1)}, IncreaseHiddenNodes)$
(4) $Model_{(k)} = SaveModel(\mathbf{A}_{(k)}, \mathbf{b}_L, \mathbf{K}_{(k)}, \boldsymbol{\beta}_{(k)})$
(5) **return** $Model_{(k)}$

Algorithm 4: Algorithm AOS-ELM sequential-hybrid drift.

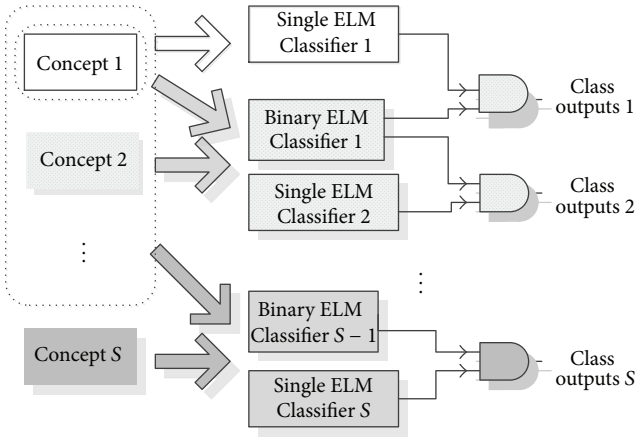


Figure 4: Hierarchical ELM ensemble for MNIST + USPS experiment. The gray shadow showed the new classifiers assembled when the new concept presented.

shuffling the composition of $\mathbf{C}_1$ and $\mathbf{C}_2$. We set the sequential training flow to be the following drift equation:

(a) Experiment 1: virtual drift: MNIST$[X_{\text{grey}}] \overset{\gg}{}_{\text{VD}}$ MNIST$[X_{\text{grey}} X_{\text{HOG}}]$,

(b) Experiment 2: real drift: for recurring context, $\mathbf{C}_1 \overset{\gg}{}_{\text{RD}} \text{shuffled}(\mathbf{C}_1, \mathbf{C}_2)$; for sudden drift: $\mathbf{C}_1 \overset{\gg}{}_{\text{RD}} \mathbf{C}_2$,

(c) Experiment 3: hybrid drift: $\mathbf{C}_1 \overset{\gg}{}_{\text{HD}} \text{shuffled}(\mathbf{C}_3, \mathbf{C}_4)$,

(d) Experiment 4: MNIST + USPS transfer learning: $\mathbf{C}_5 \overset{\gg}{}_{\text{RD}} \mathbf{C}_6$.

(5) We measured the performance based on Table 2(c). The testing accuracy and Cohen's Kappa are to show the quantitative measurement. The predictive accuracy is to demonstrate the trend in a line chart. The sudden drift performance is based on the forgetting capability that compared the testing accuracy of the latest concept against all the previous concepts.

(6) We compared the AOS-ELM performance with nonadaptive online sequential and offline version of ELM classifier. The performance expectation of sequential version classifier is to approximate the offline version of the classifier (desiderata for online classifiers [22]). We also compared with adaptive ELM ensemble method (see Figure 4). We designed the hierarchical ensemble using two models of ELM classifier with different roles (see Figure 4). The first role is a binary classifier that acts as a director based on one against all (OAA) classification. The binary classifier needs all sequential training data to be recalled (full memory). Another role is the data classifier. This ensemble requires total $2S - 1$ classifiers for S concepts, thus, not effective for consecutive concept drift case, for example, SEA concepts. The ensemble also applied outdated classifier pruning when the ensemble detects that the previous attributes need to be replaced.

Table 2: Data set dimension, quantity, evaluation method, and performance measurement.

(a) Data set dimension and quantity

Data set	Concepts	Inputs	Outputs	Quantity (×concepts)
SEA	4	3	2	20000 (×4)
STAGGER	3	9	2	4400 (×3)
MNIST	2	784, 865	10	70000 (×2)
USPS	1	865	36	48908 (×1)

(b) Evaluation method

Data set	Evaluation method	Training	Testing
SEA	5-fold cross-validation	16000 (×4)	4000 (×4)
STAGGER	5-fold cross-validation	3520 (×3)	880 (×3)
MNIST	Holdout (10x trials)	60000 (×2)	10000 (×2)
USPS	Holdout (10x trials)	35050	13858

(c) Performance measurements

Measure	Specification
Accuracy	The accuracy of classification in % from # correctly classified/# total instances
Predictive accuracy	The accuracy measurement of the future sequential training data [20]
Testing accuracy	The accuracy measurement of the testing data set excluded from the training
Forgetting capability	The testing accuracy differences between the current concept with the previous concepts
Cohen's kappa and kappa error	The statistic measurement of interrater agreement for categorical items

4.2. SEA and STAGGER Concepts Result. We addressed the question whether nonadaptive OS-ELM and CEOS-ELM with δL increase could handle the concept drift situation. We compared between AOS-ELM with no δL increase (AOS-ELM1) and with δL increase (AOS-ELM2). We used 5-fold cross-validation and compared between NORM and ROS parameter. For SEA, parameters $L_0 = 3000$ and $\delta L = 500$ increase per drift. For STAGGER, parameters $L_0 = 9$ and $\delta L = 5$ hidden nodes increase per drift.

The AOS-ELM has better accuracy with better recovery time (see Tables 3(a) and 3(b)) than CEOS-ELM, whereas nonadaptive OS-ELM fails (see Figure 5). The AOS-ELM2 improved the forgetting capability better than AOS-ELM1. In comparison with Kolter and Maloof result using dynamically weighted majority (DWM) of naive Bayes (DWM-NB) for SEA, AOS-ELM result is near to the DWM result. Comparison with inducing decision trees (DWM-ITI) for STAGGER [20], AOS-ELM outperformed DWM (see Tables 3(a) and 3(b)).

Table 3: Average testing accuracy in % for each concept between OS-ELM, CEOS-ELM, AOS-ELM1, and AOS-ELM2.

(a) Testing accuracy in % for SEA with $\mathbf{C}_1 \overset{\gg}{\underset{RD}{}} \mathbf{C}_2 \overset{\gg}{\underset{RD}{}} \mathbf{C}_3 \overset{\gg}{\underset{RD}{}} \mathbf{C}_4$

Method	Parameter	$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{C}_3$	$\mathbf{C}_4$
OS-ELM	NORM	89.48 ± 0.33	86.47 ± 0.37	83.41 ± 0.93	85.32 ± 0.45
	ROS	89.51 ± 0.37	86.43 ± 0.38	83.49 ± 0.95	85.31 ± 0.49
CEOS-ELM	NORM	82.40 ± 0.27	89.33 ± 0.43	75.80 ± 0.87	90.30 ± 0.35
	ROS	84.54 ± 0.59	89.96 ± 0.59	77.97 ± 1.05	90.26 ± 0.69
AOS-ELM1	NORM	89.84 ± 0.28	90.03 ± 0.25	89.67 ± 0.61	**90.33** ± 0.39
	ROS	89.76 ± 0.34	90.02 ± 0.25	89.76 ± 0.55	**90.34** ± 0.38
AOS-ELM2	NORM	50.58 ± 1.18	50.71 ± 1.19	48.50 ± 10.10	**90.34** ± 0.30
	ROS	65.78 ± 1.21	65.67 ± 1.19	64.09 ± 1.89	**90.14** ± 1.34

(b) Testing accuracy in % for STAGGER with $\mathbf{C}_1 \overset{\gg}{\underset{RD}{}} \mathbf{C}_2 \overset{\gg}{\underset{RD}{}} \mathbf{C}_3$

Method	Parameter	$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{C}_3$
OS-ELM	NORM	51.89 ± 3.48	81.61 ± 4.74	67.18 ± 5.80
	ROS	49.77 ± 1.96	84.16 ± 1.61	66.93 ± 2.00
CEOS-ELM	NORM	21.98 ± 1.57	53.66 ± 4.34	97.84 ± 4.32
	ROS	23.23 ± 1.93	52.11 ± 2.35	99.27 ± 1.45
AOS-ELM1	NORM	97.64 ± 1.95	**100.00** ± 0.00	**100.00** ± 0.00
	ROS	**100.00** ± 0.00	**100.00** ± 0.00	**100.00** ± 0.00
AOS-ELM2	NORM	59.66 ± 5.65	70.91 ± 10.93	**100.00** ± 0.00
	ROS	56.20 ± 9.56	69.41 ± 14.05	**100.00** ± 0.00

4.3. Concept Drift Detection. The drift detection works based on loss estimation (see Figure 2) that compared current prediction accuracy with the previous feedback. Using similar method on [33, 34], we can evaluate the intersection point between accuracy decrease and increase in Figure 6. If the consecutive loss performance exceeded a certain threshold, then drift warning status is triggered. We measured the output performance from the new concept output and compared with the previous output. If it met certain criteria, then the new AOS-ELM is committed. Otherwise, the previous AOS-ELM is rolled back.

4.4. MNIST and MNIST + USPS Result. We measured the testing accuracy based on holdout test data by 10x experiment trials. The results are as follows.

Experiment 1 (virtual drift). The AOS-ELM of $[X_{\text{grey}} X_{\text{HOG}}]$ has Cohen's kappa of testing accuracy 95.72 (0.21)% approximated to its nonadaptive ELM and offline ELM of $[X_{\text{grey}} X_{\text{HOG}}]$ version with the same hidden nodes number $L = 2000$. It has better accuracy than single attribute $[X_{\text{grey}}]$ or $[X_{\text{HOG}}]$ only (see Table 4(b)). It proves our explanation in the theoretical background on Section 3.1.

Note. We set $L_0 = 200$ for $[X_{\text{HOG}}]$ ELM based on the same ratio between number of input nodes with hidden nodes of $[X_{\text{grey}}]$ ELM.

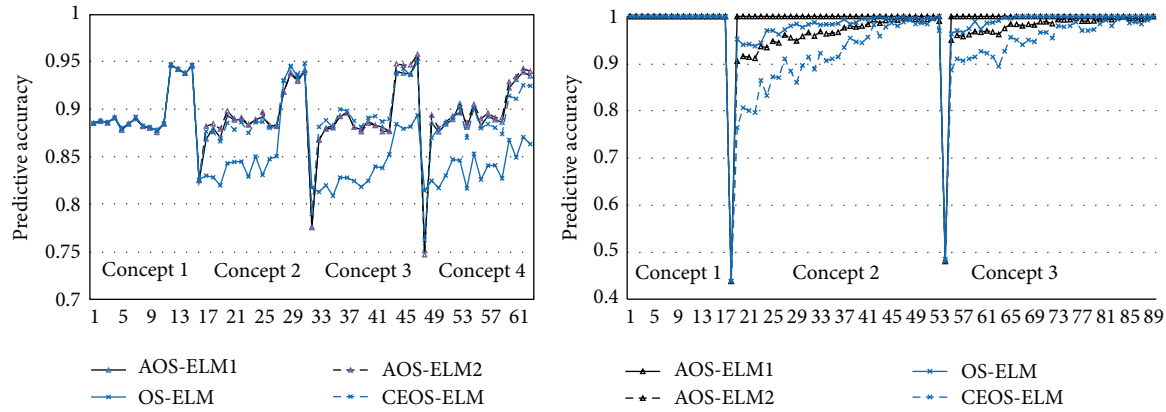
Experiment 2 (real drift). The final result is shown in Table 5(b): the AOS-ELM has better Cohen's kappa performance for all concepts than ELM ensemble and little exceeds its nonadaptive and offline ELM (Table 5(b)).

As in the split composition, the AOS-ELM with δL increase has better performance in forgetting capability than the AOS-ELM with no δL increase (see Table 8(b)).

Experiment 3 (hybrid drift). The final result is shown in Table 5(c): the AOS-ELM has better Cohen's kappa performance for HD than ELM ensemble and approximates to its nonadaptive and offline ELM.

Experiment 4 (MNIST + USPS transfer learning). The AOS-ELM has better Cohen's kappa performance for both numeric and alphabet concepts than ELM ensemble (see Table 5(d)) and approximates to its nonadaptive and offline ELM. The AOS-ELM shows better recovery time than ELM ensemble in Figure 7.

4.5. The Effect of Hidden Nodes Increase. The initial size of hidden nodes L_0 selection is important to have good generalization performance. Researches [3, 6] suggested for the hidden nodes size to be at minimum equal to the rank value of training data. However, in a data stream, it is hard to determine a fixed number of hidden nodes following that suggestion. The larger L_0 requires more computation resources



(a) The SEA concept (b) The STAGGER concept

Figure 5: Predictive accuracy of AOS-ELM1 (black line △) and AOS-ELM2 (black dashes △) compared with OS-ELM (blue line ×) and CEOS-ELM (blue dashes ×).

Table 4: Average testing accuracy and Cohen's kappa in % for MNIST VD experiment (other ELM parameters are same: ROS, $\delta L = 0$, $\delta N = 1000$) with 10x trials.

(a) Benchmark result, nonadaptive OS-ELM and offline ELM

Performance	ELM method	$[X_{\text{grey}}]$ ($L = 2000$)	$[X_{\text{HOG}}]$ ($L = 200$)	$[X_{\text{grey}} X_{\text{HOG}}]$ ($L = 2000$)
Testing accuracy	OS-ELM	95.32 ± 0.12	94.64 ± 0.15	**96.86** ± 0.13
	Offline ELM	95.33 ± 0.13	94.66 ± 0.15	**96.85** ± 0.06
Cohen's kappa	OS-ELM	94.80 (0.24)	94.04 (0.25)	**96.51 (0.19)**
	Offline ELM	94.81 (0.23)	94.06 (0.25)	**96.50 (0.19)**

(b) VD experiment, AOS-ELM ($L = 2000$)

Drift	Testing accuracy	Cohen's kappa
MNIST $[X_{\text{grey}}] \overset{\gg}{\text{VD}}$ MNIST $[X_{\text{grey}} X_{\text{HOG}}]$	**96.15** ± 0.08	**95.72 (0.21)**

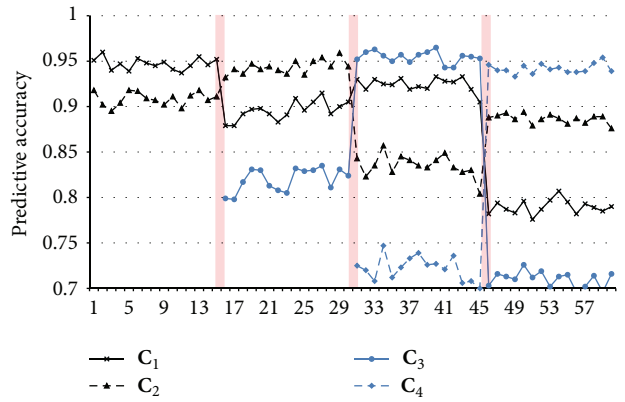


Figure 6: Predictive accuracy of AOS-ELM in SEA for each concept with m output (see Figure 3). We can consider the intersection point between accuracy decrease of previous concept and accuracy increase of current concept as change point (displayed as thin vertical shadow line).

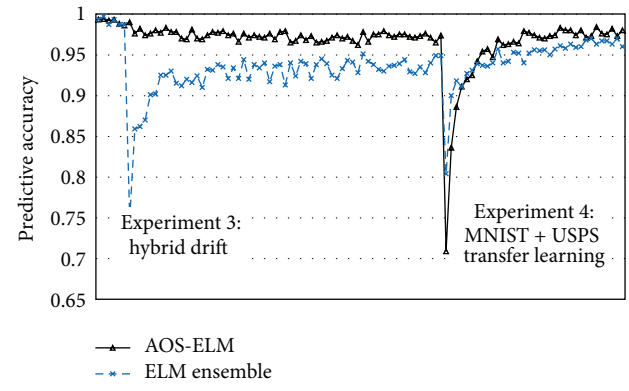


Figure 7: Predictive accuracy of AOS-ELM (black line) over sequential data for MNIST + USPS compared with ELM ensemble (blue dash line).

and processing time, probably not giving a significant result at the end. Thus, we have a requirement to increase δL in sequential stage [9].

The experiment result in Table 6 shows that the performance improved when certain hidden nodes size increases. We used different initial hidden nodes size (L_0) condition: 2000, 666 (the rank value of initial training data), and 713 (the rank value of total training data). We used also different

Table 5: Average testing accuracy and Cohen's kappa in % for MNIST RD, HD, and MNIST + USPS transfer learning experiment (other ELM parameters are same: ROS, $L = 2000$, $\delta L = 0$, $\delta N = 1000$) with 10x trials.

(a) Benchmark result, nonadaptive OS-ELM and offline ELM

Source	Class	Testing accuracy		Cohen's kappa	
		OS-ELM	Offline ELM	OS-ELM	Offline ELM
MNIST $[X_{\text{grey}}]$	(1–6)	95.99 ± 0.15	96.00 ± 0.14	95.21 (0.30)	95.22 (0.30)
	(7–10)	94.30 ± 0.22	94.32 ± 0.19	92.50 (0.48)	92.53 (0.48)
MNIST $[X_{\text{grey}}X_{\text{HOG}}]$	(1–6)	97.59 ± 0.11	97.49 ± 0.09	97.10 (0.23)	97.00 (0.24)
	(7–10)	95.76 ± 0.26	95.87 ± 0.12	94.40 (0.42)	94.55 (0.42)
MNIST + USPS $[X_{\text{grey}}X_{\text{HOG}}]$	(1–10)	96.01 ± 0.10	96.08 ± 0.08	95.56 (0.02)	95.65 (0.02)
	(A–Z)	99.94 ± 0.02	99.94 ± 0.02	99.94 (0.02)	99.93 (0.02)

(b) RD experiment, ELM ensemble (3 classifiers, full memory) versus AOS-ELM

Source	Concept	Testing accuracy		Cohen's kappa	
		ELM ensemble	AOS-ELM	ELM ensemble	AOS-ELM
MNIST $[X_{\text{grey}}]$	$\mathbf{C}_1$ (1–6)	94.58 ± 0.17	**96.09** ± 0.12	93.54 (0.35)	**95.10 (0.31)**
	$\mathbf{C}_2$ (7–10)	91.60 ± 0.29	**94.34** ± 0.16	89.04 (0.57)	**92.56 (0.48)**

(c) HD experiment, ELM ensemble (3 classifiers, full memory, outdated classifier pruning) versus AOS-ELM

Source	Concept	Testing accuracy		Cohen's kappa	
		ELM ensemble	AOS-ELM	ELM ensemble	AOS-ELM
MNIST $[X_{\text{grey}}]$	$\mathbf{C}_3$ (1–6)	94.48 ± 0.33	**97.01** ± 0.18	93.42 (0.35)	**96.42 (0.26)**
MNIST $[X_{\text{grey}}X_{\text{HOG}}]$	$\mathbf{C}_4$ (7–10)	92.29 ± 0.36	**96.05** ± 0.19	89.95 (0.55)	**94.78 (0.40)**

(d) MNIST + USPS experiment, ELM ensemble (5 classifiers, full memory, outdated classifier pruning) versus AOS-ELM

Source	Concept	Testing accuracy		Cohen's kappa	
		ELM ensemble	AOS-ELM	ELM ensemble	AOS-ELM
MNIST $[X_{\text{grey}}X_{\text{HOG}}]$	$\mathbf{C}_5$ (1–10)	88.17 ± 11.06	**95.91** ± 0.12	86.94 (0.33)	**95.46 (0.22)**
USPS $[X_{\text{grey}}X_{\text{HOG}}]$	$\mathbf{C}_6$ (A–Z)	99.80 ± 0.05	**99.95** ± 0.03	99.79 (0.40)	**99.95 (0.02)**

conditions of hidden nodes increase (δL) by using ROS parameters on the drift event: 0 (no increase), 500, 1000, and 2000. However, the larger L_0 has better influence than δL increase.

We studied the effect of hidden nodes increased in the sequential phase as follows.

(1) "Underfitting" Condition. "Underfitting" is the condition when the model does not fit the data well enough which makes nonconvergence. Based on an empirical experiment with δL increase in the sequential phase on Table 7, we investigated particular condition when the AOS-ELM classifier has a bad result. We realized that the ELM performance is dependent upon finding general matrix inverse of $\mathbf{H}$. Based on orthogonal projection method in CEOS-ELM, we can employ the rank value of $\widehat{\mathbf{P}}$ as evaluation parameter to detect "underfitting".

The $\widehat{\mathbf{P}}$ is approximation to matrix $(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}$. The full rank of $\mathbf{P}_{L\times L}$ is ideally equal to L. However, certain condition in the sequential training, for example, poor training data or poor learning parameter selection, may cause the diagonal squared matrix $\mathbf{P}$ to be less diagonalizable [35], thus not full rank anymore.

In the sequential learning, we can compare Rank($\widehat{\mathbf{P}}$) before and after hidden nodes increase. The expected result is positive increment. If the rank value becomes lower after hidden nodes increase, then it has a higher probability for "underfitting" condition to occur. Rank($\widehat{\mathbf{P}}$) is determined by the block size of training data, the number of hidden nodes increment, the c scalar selection in ROS parameter, the activation function, input weight, and bias random assignment method. In this experiment, we focused on the block size, the number of hidden nodes, and c scalar selection. Using Rank($\widehat{\mathbf{P}}$) as evaluation parameter is more efficient because we do not need to compute $\boldsymbol{\beta}$.

(2) Sudden Drift. In Tables 8(a) and 8(b), the hidden nodes increase can improve the forgetting capability on sudden drift (it reduced the accuracy of the outdated concept).

In CEOS-ELM, when δL increases in the same time with drift, it makes $\begin{bmatrix} \mathbf{H}_1 & \mathbf{0} \\ \mathbf{H}_2 & \Delta\mathbf{H}_2 \end{bmatrix}$ and the new concept target $\begin{bmatrix} 0 \\ \mathbf{t}_2 \end{bmatrix} \in \mathbf{T}_2$ in split composition, while previous concept $\begin{bmatrix} \mathbf{t}_1 \\ 0 \end{bmatrix} \in \mathbf{T}_1$. Thus, in the process of finding $\widehat{\boldsymbol{\beta}}$ it becomes simplified because $[\mathbf{H}_2\Delta\mathbf{H}_2]$ is partially trained by $\mathbf{t}_2$ only and not by $\mathbf{t}_1$. Thus, it reduced the generalization capability of $[\mathbf{H}_2\Delta\mathbf{H}_2]$ to recognize $\mathbf{T}_1$ problem.

Table 6: Testing accuracy in Cohen's kappa (kappa error) in % AOS-ELM MNIST for different L_0 and δL.

Scenario	L_0	$\delta L = 0$	$\delta L = 500$	$\delta L = 1000$	$\delta L = 2000$
VD	2000	95.92 (0.21)	96.37 (0.20)	96.83 (0.18)	**96.89 (0.18)**
	666	93.10 (0.27)	95.18 (0.23)	**96.18 (0.20)**	95.60 (0.22)
	713	93.30 (0.26)	95.31 (0.22)	**96.28 (0.20)**	96.09 (0.20)
RD	2000	94.71 (0.24)	94.93 (0.23)	95.39 (0.22)	**95.42 (0.22)**
	630	91.3 (0.30)	91.67 (0.29)	93.61 (0.26)	**94.04 (0.25)**
	713	91.71 (0.29)	92.70 (0.28)	93.82 (0.25)	**94.23 (0.25)**

Table 7: Predictive accuracy performance in AOS-ELM for MNIST using different parameters and Rank($\widehat{\mathbf{P}}$) before and after δL increase. Each experiment is repeated 100x trials to get the probability of predictive accuracy ≤ 50%.

Scenario	L_0	δL	Batch size	c	Before	After	Pred. acc. ≤ 50%
RD	630	500	1000	10	630	1130	0%
	630	500	500	10	1130	1122	**7**%
	630	500	100	10	1130	614	**5**%
	630	500	10	10	640	640	0%
	630	100	1000	10	630	730	0%
	630	100	500	10	730	730	0%
	630	100	100	10	730	730	0%
	630	100	10	10	640	640	0%
	2000	1000	100	5	2000	1868	**3**%
	2000	1000	100	1	2000	1947	**16**%
	2000	1000	100	0.5	2000	1946	**17**%
	2000	1000	100	0.05	2000	2100	0%
VD	666	500	500	5	666	1166	0%
	666	500	100	5	666	1166	0%
	666	100	500	5	666	766	0%
	666	100	100	5	666	766	0%
	666	50	500	5	666	716	0%
	666	50	100	5	666	716	0%

5. AOS-ELM in Regression

We can use the similar real drift scenario with output marginalization and output amplification to solve concept drift problem in regression. In this experiment, we used AOS-ELM with single input node and single output node per concept. We defined the following concept as

(i) $\mathbf{C}_1$ is *sinc* function with 50000 training/5000 testing,

(ii) $\mathbf{C}_2$ is *sinus* function with 50000 training/5000 testing,

(iii) $\mathbf{C}_3$ is *gaussian* function with 50000 training/5000 testing.

The sequential experiments are following drift equations:

(1) For Experiment 1, $\mathbf{C}_1 \overset{\gg}{\underset{\text{RD}}{}} \mathbf{C}_2$,

(2) For Experiment 2, $\mathbf{C}_1 \overset{\gg}{\underset{\text{RD}}{}} \mathbf{C}_2 \overset{\gg}{\underset{\text{RD}}{}} \mathbf{C}_3$.

We presented the result on Figures 8–10 to compare the performance of each concept at the end of each training experiment. Our objective is to show the AOS-ELM regression capability to keep the previous regression concept knowledge. We select the constant value giving the best regression result of each concept. The AOS-ELM has $L_0 = 100$, $\delta L = 0$, and *sigmoid* function. More drifts occurring will weaken the older concepts. Thus, the AOS-ELM needs larger amplifier constant value.

6. Simulation in Big Data Stream: Intrusion Detection System (IDS) KDD Cup 1999

IDS is a network security technology that scans any network packet traffic to detect any potential exploits, then sending the alarm or taking some active action to Intrusion Prevention System. Some machine learning methods have been applied with the hope of improving detection rates and adaptive capability [36].

In this experiment, we used KDD Cup 1999 Competition data set. The full data set had 4898431 network packets grouped to be 23 classes (One normal class and 22 attack names based on a signature-based detection) [37]. The data set has a control information (CI) header for delivering the data in numerical and multicategorical values as features. We focused on service names (IP ports) attributes because they are specific differentiators for applications. The CI and the number of attack classes are not stationary. We analyzed

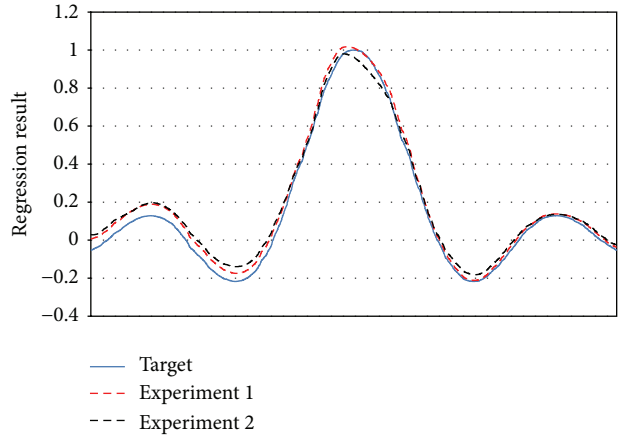


Figure 8: Regression result of $\mathbf{C}_1$ for Experiment 1 (red dash line) using constant value 4 and Experiment 2 (black dash line) using constant value 8.25.

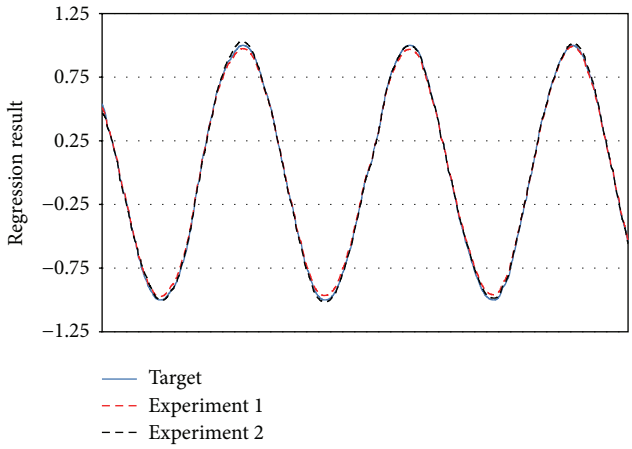


Figure 9: Regression result of $\mathbf{C}_2$ for Experiment 1 (red dash line) using constant value 1.3 and Experiment 2 (black dash line) using constant value 3.

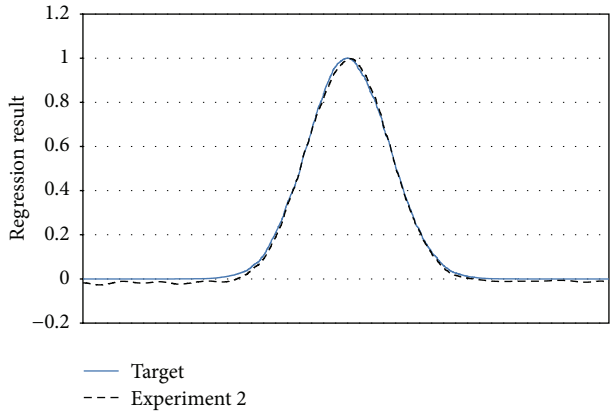


Figure 10: Regression result of $\mathbf{C}_3$ for Experiment 2 (black dash line) using constant value 1.85.

Table 8: Average testing accuracy in % for RD experiment. For SEA, we started from $\mathbf{C}_2$ to $\mathbf{C}_4$. For MNIST, we started from $\mathbf{C}_1$ to $\mathbf{C}_2$.

(a) Sudden drift effect caused by split training composition with hidden nodes δL increase

Data	AOS-ELM	δL	Concept	Testing accuracy
SEA	$L_0 = 3000$	0	$\mathbf{C}_2$	**90.00** ± 0.59
			$\mathbf{C}_4$	**90.24** ± 0.61
		500	$\mathbf{C}_2$	66.29 ± 1.12
			$\mathbf{C}_4$	**90.12** ± 0.52
MNIST	$L_0 = 2000$	0	$\mathbf{C}_1$	**96.42** ± 0.21
			$\mathbf{C}_2$	**93.68** ± 0.23
		500	$\mathbf{C}_1$	17.59 ± 0.98
			$\mathbf{C}_2$	**97.08** ± 0.15

(b) MNIST RD simulation: the effect of hidden nodes δL increase for split and shuffled training composition ($L_0 = 2000$)

Data	δL	Composition	$\mathbf{C}_1$	$\mathbf{C}_2$
MNIST	0	Split	**96.42** ± 0.21	**93.68** ± 0.23
		Shuffled	**96.09** ± 0.12	**94.34** ± 0.16
	500	Split	17.59 ± 0.98	**97.08** ± 0.15
		Shuffled	**96.53** ± 0.12	**94.29** ± 0.25
	1000	Split	8.65 ± 1.13	**97.64** ± 0.18
		Shuffled	**96.74** ± 0.14	**94.78** ± 0.10

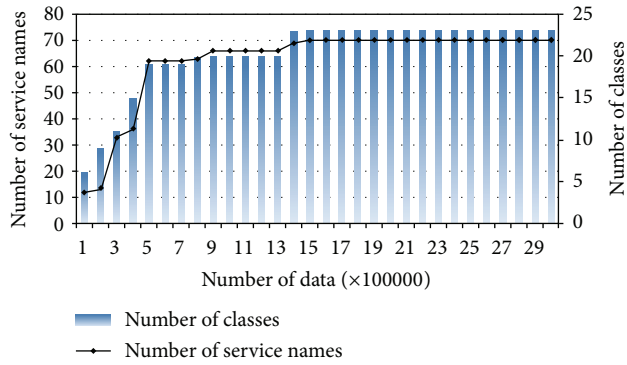


Figure 11: The changes on service names and number of classes following the streaming data.

the data set for the growing of service names and the number of class attack in the whole data set on Figure 11. The challenge in IDS data set is imbalanced data between the classes. The highest number of data is for "normal" class, and the lowest number is for "spy" class (only 2 packets). To simplify the experiment, we use oversampling by adding more data based on the random normal distribution of packet signatures and under sampling approaches by dropping some samples randomly.

Based on the growing of service names and the number of classes analysis (see Figure 11), we designed one drift scenario based on two concepts (Table 9(b)). $\mathbf{C}_1$ has ten classes and 37 service names and $\mathbf{C}_2$ has 23 classes and 70 service names. Total training data for each concept is 920000 packets. There is no data repetition from the previous event, except at the end

Table 9: Performance comparison of AOS-ELM and nonadaptive OS-ELM to validation data set $\mathbf{C}_2$ in %.

(a) Benchmark result, nonadaptive OS-ELM

Concept	Parameters	Testing accuracy	Cohen's kappa
$\mathbf{C}_1$	OS-ELM	50.87 ± 0.01	45.81 (0.54)
$\mathbf{C}_2$	OS-ELM	94.58 ± 0.05	94.03 (0.24)

(b) Performance result on the drift event, AOS-ELM

Drift	Parameters	Testing accuracy	Cohen's kappa
$\mathbf{C}_1 \overset{\ggg}{\text{HD}} \mathbf{C}_2$	AOS-ELM1	92.18 ± 2.73	91.38 (0.29)
$\mathbf{C}_1 \overset{\ggg}{\text{HD}} \mathbf{C}_2$	AOS-ELM2	94.64 ± 0.06	94.10 (0.24)
End of full $\mathbf{C}_2$	AOS-ELM1	93.45 ± 1.18	92.78 (0.27)
End of full $\mathbf{C}_2$	AOS-ELM2	94.57 ± 0.11	94.02 (0.25)

of $\mathbf{C}_2$ sequential training. The composition between $\mathbf{C}_1/\mathbf{C}_2$ on HD event is 230000/690000. The validation data set of $\mathbf{C}_2$ is selected from all packets from minority classes and randomly selected original majority classes (10422 packets). We used holdout method with 5x trials. We used AOS-ELM1 for $\delta L = 0$ and AOS-ELM2 for $\delta L = 500$ (other ELM parameters are same: $L_0 = 1000$, NORM, *sig*). The AOS-ELM result in this experiment can approximate the nonadaptive OS-ELM on $\mathbf{C}_2$ (see Table 9(b)).

7. Challenges and Future Research

Based on AOS-ELM experiments, we face some challenges, which are as follows:

(i) We need to investigate the optimum transition space that minimizes the gap to the new concept learning model. In certain case, the AOS-ELM may have the "underfitting" condition and require larger training data to achieve the new convergence.

(ii) We need to check the consistency of AOS-ELM for different pseudoinverse methods (e.g., Greville's method [23]).

We suggest some ideas for AOS-ELM future researches as follows:

(i) The need for transfer learning to solve big data problem when the distribution data changes.

(ii) The AOS-ELM integration with other ELM methods, for example, Weighted OS-ELM for imbalanced learning [38], ELM Autoencoder (ELM-AE) [39], and Stacked ELM [40].

(iii) A detailed systematic explanation based on rule extraction [41] for AOS-ELM in handling adaptive environment.

8. Conclusion

The proposed method gives better adaptive capability than nonadaptive OS-ELM and CEOS-ELM in terms of retaining the recognition performance when handling concept drifts. It uses a simple line of code easy to deploy especially for consecutive drifts, compared with adaptive ensemble methods. While most adaptive classifiers work differently for each of virtual, real drift, and hybrid drift scenarios, the AOS-ELM tackles those drifts through simple block matrix reconstruction and rank evaluation.

AOS-ELM satisfied the requirement criteria in terms of accuracy, simplicity, speed, and flexibility. However, in certain VD and HD cases, the AOS-ELM accuracy may not exceed the nonadaptive sequential ELM, which include the future training data. In RD cases, the AOS-ELM has better accuracy. In a real data implementation, the nonadaptive ELM is better and preferred when we know exactly the future behavior of data. However, we can not predict it precisely. We believe using larger training data, the AOS-ELM performance will approximate the expected value of nonadaptive sequential ELM or offline ELM, which use the future training data. The AOS-ELM can also add learning adaptation function to the previous offline learning model. It makes AOS-ELM an excellent choice for the unpredictable situation.

The AOS-ELM tackles sudden drift change type as well as recurrent context change type. The output marginalization strategy is implemented by simply shifting the output nodes belonging to the latest concept. The AOS-ELM does need to increase the hidden nodes to improve the forgetting capability for sudden drift change type. To make sure of the convergence to the expected learning model, we proposed the rank value of the pseudoinverse autocorrelation hidden nodes matrix as evaluation parameter to prevent "underfitting" condition that makes the accuracy performance dropped.

We can consider the AOS-ELM as another type of ELM ensemble formation using shared and interconnected hidden nodes between ensemble members. We can implement the AOS-ELM in similar fashion compared to the ELM ensemble for adaptive learning scheme, but with better performance, simplicity, and more resource efficiency. However, the AOS-ELM does have some drawbacks. Any hidden node changes could impact all notions.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work is supported by Higher Education Center of Excellence Research Grant funded by Indonesia Ministry of Research, Technology and Higher Education (Contract no. 1068/UN2.R12/HKP.05.00/2016).

References

[1] J. A. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, article 44, pp. 1–37, 2014.

[2] P. B. Dongre and L. G. Malik, "A review on real time data stream classification and adapting to various concept drift scenarios," in *Proceedings of the 4th IEEE International Advance Computing*

Conference (IACC '14), pp. 533–537, Gurgaon, India, February 2014.

[3] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[4] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[5] G.-B. Huang, "An insight into extreme learning machines: random neurons, random features and kernels," *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.

[6] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: a review," *Neural Networks*, vol. 61, pp. 32–48, 2015.

[7] G.-B. Huang, "What are extreme learning machines? filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle," *Cognitive Computation*, vol. 7, no. 3, pp. 263–278, 2015.

[8] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[9] Y. Lan, Y. C. Soh, and G.-B. Huang, "A constructive enhancement for Online Sequential Extreme Learning Machine," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '09)*, pp. 1708–1713, Atlanta, Ga, USA, June 2009.

[10] L. I. Kuncheva, "Classifier ensembles for detecting concept change in streaming data: overview and perspectives," in *Proceedings of the 2nd Workshop SUEMA (ECAI '08)*, pp. 5–10, Patras, Greece, 2008.

[11] T. G. Dietterich, "Ensemble methods in machine learning," in *Proceedings of the 1st International Workshop on Multiple Classifier Systems (MCS '00)*, pp. 1–15, Springer, 2000.

[12] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers and Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[13] I. Žliobaitė, "Learning under concept drift: an overview," *CoRR-Computing Research Repository*, 2010.

[14] A. Budiman, M. I. Fanany, and C. Basaruddin, "Constructive, robust and adaptive OS-ELM in human action recognition," in *Proceedings of the International Conference on Industrial Automation, Information and Communications Technology (IAICT '14)*, pp. 39–45, Bali, Indonesia, August 2014.

[15] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[16] Q. Yang, "Transfer learning beyond text classification," in *Proceedings of the 1st Asian Conference on Machine Learning (ACML '09)*, Advances in Machine Learning, pp. 10–22, Springer, Berlin, Germany, 2009.

[17] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010, http://yann.lecun.com/exdb/mnist/.

[18] S. Roweis, Data for matlab hackers handwritten digits, http://www.cs.nyu.edu/~roweis/data.html.

[19] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01)*, 2001.

[20] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: an ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.

[21] J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM '07)*, 2007.

[22] L. Kuncheva, "Classifier ensembles for changing environments," in *Multiple Classifier Systems*, F. Roli, J. Kittler, and T. Windeatt, Eds., vol. 3077 of *Lecture Notes in Computer Science*, pp. 1–15, Springer, Berlin, Germany, 2004.

[23] A. van Schaik and J. Tapson, "Online and adaptive pseudoinverse solutions for ELM weights," *Neurocomputing*, vol. 149, pp. 233–238, 2015.

[24] K. Cao, G. Wang, D. Han, J. Ning, and X. Zhang, "Classification of uncertain data streams based on extreme learning machine," *Cognitive Computation*, vol. 7, no. 1, pp. 150–160, 2015.

[25] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Springer, New York, NY, USA, 1996.

[26] C. M. Grinstead and J. L. Snell, *Introduction to Probability*, AMS, 2003, http://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/book.html.

[27] L. C. A. Corsten, "Statistical methods: the geometric approach (David J. Saville and Graham R. Wood)," *SIAM Review*, vol. 34, no. 3, pp. 506–508, 1992.

[28] R. Strack, *Geometric approach to support vector machines learning for large datasets [Ph.D. thesis]*, Virginia Commonwealth University, Richmond, Va, USA, 2013.

[29] A. Iosifidis, "Extreme learning machine based supervised subspace learning," *Neurocomputing*, vol. 167, pp. 158–164, 2015.

[30] O. Ludwig Junior, D. Delgado, V. Gonçalves, and U. Nunes, "Trainable classifier-fusion schemes: an application to pedestrian detection," in *Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems (ITSC '09)*, pp. 1–6, St. Louis, Mo, USA, October 2009.

[31] M.-T. T. Hoang, H. T. Huynh, N. H. Vo, and Y. Won, "A robust online sequential extreme learning machine," in *Proceedings of the 4th International Symposium on Neural Networks: Advances in Neural Networks*, pp. 1077–1086, Springer, Berlin, Germany, 2007.

[32] A. Dries and U. Rückert, "Adaptive concept drift detection," *Statistical Analysis and Data Mining*, vol. 2, no. 5-6, pp. 311–327, 2009.

[33] K. Nishida and K. Yamauchi, "Adaptive classifiers-ensemble system for tracking concept drift," in *Proceedings of the 6th International Conference on Machine Learning and Cybernetics (ICMLC '07)*, pp. 3607–3612, Hong Kong, August 2007.

[34] C. Alippi, G. Boracchi, and M. Roveri, "Just-in-time classifiers for recurrent concepts," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 4, pp. 620–634, 2013.

[35] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Md, USA, 3rd edition, 1996.

[36] S. K. Wagh, V. K. Pachghare, and S. R. Kolhe, "Survey on intrusion detection system using machine learning techniques," *International Journal of Computer Applications*, vol. 78, no. 16, pp. 30–37, 2013.

[37] J. Gama, "Knowledge discovery from ubiquitous streams—datasets for concept drift," http://www.liaad.up.pt/kdus/products/datasets-for-concept-drift.

[38] B. Mirza, Z. Lin, and K.-A. Toh, "Weighted online sequential extreme learning machine for class imbalance learning," *Neural Processing Letters*, vol. 38, no. 3, pp. 465–486, 2013.

[39] N. Zhang, S. Ding, and Z. Shi, "Denoising Laplacian multi-layer extreme learning machine," *Neurocomputing*, vol. 171, pp. 1066–1074, 2016.

[40] H. Zhou, G.-B. Huang, Z. Lin, H. Wang, and Y. C. Soh, "Stacked extreme learning machines," *IEEE Transactions on Cybernetics*, vol. 45, no. 9, pp. 2013–2025, 2015.

[41] N. Barakat and A. P. Bradley, "Rule extraction from support vector machines: a review," *Neurocomputing*, vol. 74, no. 1–3, pp. 178–190, 2010.