Adaptive Online Sequential ELM for Concept Drift Tackling

Arif Budiman, Mohamad Ivan Fanany, Chan Basaruddin,

Machine Learning and Computer Vision Laboratory
Faculty of Computer Science, Universitas Indonesia

* arif.budiman21@ui.ac.id

Abstract

A machine learning method needs to adapt to over time changes in the environment. Such changes are known as concept drift. One approach to concept drift handling is by feeding the whole training data set once again into a learning machine for retraining. Another approach is by rebuilding an ensemble classifiers to adapt to a new training data set. In either approach, retraining or rebuilding classifiers are expensive and not practical. In this paper, we propose an enhancement of Online-Sequential Extreme Learning Machine (OS-ELM) and its variant Constructive Enhancement OS-ELM (CEOS-ELM) by adding an 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 both drifts occurred at the same time (hybrid drift). The AOS-ELM also works well for sudden drift as well as 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 various public dataset widely used for concept drift verification from SEA and STAGGER; and other public datasets such as MNIST and USPS. Experiments show that our method gives higher kappa value compared to the multi-classifier 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 to take the rank of the pseudo inverse matrix as an indicator parameter to detect 'under-fitting' condition.

Keywords — adaptive, concept drift, extreme learning machine, online sequential.

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 is not only continuously generated but also keep changing. This challenging issue is known as concept drift [12], in which the statistical properties of the input attributes and target classes shifted over time. Such shifts can make the trained model becoming less accurate. 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 [9]. 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. [21], [20], [18], [17], [19]. 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) [28] and Constructive Enhancement OS-ELM (CEOS-ELM) [26]. 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) [25]. Our scope of attribute changes discussed in this paper is on the feature space concatenation that widely used in data fusion, kernel fusion, and ensemble learning [8] and not on the feature selection (irrelevant features removal) methods [5]. 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 [3]. 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 [17], [41] 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 denote the training data from different \( S \) concepts (sources or contexts), using the symbol \( X_s \) for training data and \( T_s \) for target data. We used the subscript font without parenthesis to show the source number.

3. We denote the drift event using the symbol \( \triangleright V_D \), where the subscript font shows the drift type. E.g., the Concept 1 has virtual drift event to be replaced by Concept 2 (Sudden drift) : \( C_1 \triangleright V_D C_2 \). The Concept 1 has real drift event to be replaced by Concept 1 and Concept 2 recurrently (Recurrent context) in the shuffled composition : \( C_1 \triangleright R_D \text{shuffled}(C_1, C_2) \).

4. 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.

5. We elaborated how the AOS-ELM for transfer learning using 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 [32]. Concept drift focuses on the time-varying domain with a small number of current data available. In contrast, transfer learn...
7. We denote the drift event using the symbol $\gg_{VD}$, where the subscript font shows the drift type. E.g., the Concept 1 has virtual drift event to be replaced by Concept 2 (Sudden drift) : $C_1 \gg_{VD} C_2$. The Concept 1 has real drift event to be replaced by Concept 1 and Concept 2 recurrently (Recurrent context) in the shuffled composition : $C_1 \gg_{RD\text{ shuffled}}(C_1, C_2)$. The example of transfer learning by using HD strategy is the transition from different data set sources but still related and has the same purpose. In this paper, we discussed the transfer learning on numeric handwritten MNIST [27] to alpha-numeric handwritten USPS [33] recognition.

8. 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.

9. 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 completed. We applied this assessment parameter actually to prevent 'under-fitting' 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 1 explains some issues and challenges in concept drift, the background of ELM, and ELM in sequential learning. Section 2 presents the background theory and algorithm derivation of the proposed method. In section 3 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) [35] and STAGGER [23], which are commonly used as benchmark in sequential learning. The real data sets are handwritten recognition data: MNIST for numeric [27] and USPS for alpha-numeric classes [33]. We studied the effect of hidden nodes increase as one of important learning parameter in section 3.5. Section 6 discusses research challenges and future directions. The conclusion presents some highlights in Section 7.

1 Related Works

1.1 Notations

We specify the notations used throughout this article for easier understanding:

- Matrix is written in uppercase bold (e.g., $X$).
- Vector is written in lowercase bold (e.g., $x$).
- The transpose of a matrix $X$ is written as $X^T$. The pseudo-inverse of a matrix $H$ is written as $H^\dagger$.
- $f$, $g$ will be used as non-linear differentiable function (activation function), e.g., sigmoid or tanh function.
- The amount of training data is $N$. Each input data $x$ contains some $d$ attributes. The target has $m$ number of classes. An input matrix $X$ can be denoted as $X_{d \times N}$ and the target matrix $T$ as $T_{N \times m}$. 

3/26
- The hidden layer matrix is $H$. The input weight matrix is $A$. The output weight matrix is $\beta$. The matrix $\Delta H$ is the additional block portion of the matrix $H$. The matrix $K$ is the auto correlation matrix of $H^T H$. The inverse of matrix $K$ is $P$.

- $H$ can be denoted as $H_{N \times L}$. $A$ can be denoted as $A_{d \times L}$ and $\beta$ can be denoted as $\beta_{L \times m}$. $\delta L$ denotes the additional nodes number of $L$.

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

- We denote the training data from different $S$ concepts (sources or contexts), using the symbol $X_s$ for training data and $T_s$ for target data. We used the subscript font without parenthesis to show the source number.

- We denote the drift event using the symbol $\gg_{VD}$, where the subscript font shows the drift type. E.g., the Concept 1 has virtual drift event to be replaced by Concept 2 (Sudden drift) : $C_1 \gg_{VD} C_2$. The Concept 1 has real drift event to be replaced by Concept 1 and Concept 2 recurrently (Recurrent context) in the shuffled composition : $C_1 \gg_{RD shuffled}(C_1, C_2)$.

### 1.2 Concept Drift Strategies

In this section, we briefly explained the various concept drift solution strategies. Gama, et. al. [12] explained 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 $X$ as:

$$P(c|X) = P(c) \frac{P(X|c)}{P(X)}$$  

(1)

Concept drift occurred when $P(c|X)$ has changed; e.g., $\exists X : P(0)(X, c) \neq P(1)(X, c)$, where $P(0)$ and $P(1)$ are respectively the joint distribution at time $t(0)$ and $t(1)$. Gama, et. al. categorized the concept drift types as following:

1. **Real Drift (RD)** refers to changes in $P(c|X)$. The change in $P(c|X)$ may be caused by a change in the class boundary (the number of classes) or the class conditional probabilities (likelihood) $P(X|c)$. The number of classes expanded and different class of data may come alternately, known as recurrent context. A drift, where a new conditional probabilities replaces the previous conditional probabilities while the number of class remained the same, is known as sudden drift. Other terms are concept shift or conditional change [13].

2. **Virtual Drift (VD)** refers to the changes in the distribution of the incoming data (e.g. $P(X)$ 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 [13], temporary drift, or sampling shift.
Figure 1. The taxonomy quadrant of adaptive supervised learning techniques. Popular concept drift handling methods are indicated by ellipses \[41\]. Our proposed method AOS-ELM is indicated by a dark blue diamond.

Kuncheva \[24,25\] 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 re-occur 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 \[41\] proposed a taxonomy of concept drift tackling methods as shown in Fig. 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.

Žliobaitė \[41\] explained that most attention on the concept drift tackling methods are drawn to multi-classifier model selection and fusion rules, but little attention on the model construction of base classifier.

Gama, et. al. \[12\] proposed a complete online adaptive learning scheme that organized four modules: memory, change detection, learning, and loss estimation (See Fig. 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 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.

1.3 ELM in Sequential Learning

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

ELM is getting popularity thanks to its learning speed, generalization capability, and simplicity. Huang [18] 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 $H$ can be presented as the function of:

$$f_L(x) = \sum_{i=1}^{L} \beta_i H(a_i, b_i, x)$$

(2)

where $H^\dagger$ is the pseudo inverse of $H$.

$H^\dagger$ can be approximated by left pseudoinverse of $H$ as:

$$\hat{\beta} = (H^T H)^{-1} H^T T$$

(3)

We can use ridge regression or regularized least squares to be: $\hat{\beta} = (H^T H + \frac{1}{\lambda})^{-1} H^T T$.

Based on [3], Liang et al. [28] proposed online learning for ELM named OS-ELM.

If we have $\hat{\beta}(0)$ from $H(0)$ filled by the $N_0$ number of training data and $N_1$ incremental batch of data filled $H(1)$, the output weights $\hat{\beta}(1)$ are approximated by:

$$\hat{\beta}(1) = \left( \begin{bmatrix} H(0) \\ H(1) \end{bmatrix} \right)^T \left( \begin{bmatrix} H(0) \\ H(1) \end{bmatrix} \right)^{-1} \begin{bmatrix} H(0) \\ H(1) \end{bmatrix}^T \begin{bmatrix} T(0) \\ T(1) \end{bmatrix}$$

(4)

Both $H_0$ and $H_1$ have a different number of training data but have the same $L$ number of hidden nodes.

If $K = H^T H$, then we can rewrite:

$$\hat{\beta}(1) = K^{-1}(1) \begin{bmatrix} H(0) \\ H(1) \end{bmatrix}^T \begin{bmatrix} T(0) \\ T(1) \end{bmatrix}$$

(5)

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 \cite{26} has addressed this problem by adding hidden nodes in the sequential learning stage. So $H = \begin{bmatrix} H(0) & \Delta H(0) \\ H(1) & \Delta H(1) \end{bmatrix}$.

The sub-matrix $\Delta H(0)$ 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 $\Delta H(1)$ for $N_1$ data, has relation to the additional hidden nodes $\delta L_{(1)}$.

Then, we can rewrite $K_{(1)}$ with $\Delta H(1)$ as:

$$K_{(1)} = \begin{bmatrix} H(0) & 0 \\ H(1) & \Delta H(1) \end{bmatrix}^T \begin{bmatrix} H(0) & 0 \\ H(1) & \Delta H(1) \end{bmatrix}$$ (6)

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

$$\hat{\beta}_{(1)} = \hat{P}_{(1)} \begin{bmatrix} H(0) & 0 \\ H(1) & \Delta H(1) \end{bmatrix}^T \begin{bmatrix} T(0) \\ T(1) \end{bmatrix}$$ (7)

It is important to note that both OS-ELM and CEOS-ELM did not address the concept drift issue; e.g., when the number of attributes $d$ in $X_d \times N$ or the number of classes $n$ in $T_N \times m$ in data set has added. In this paper, we categorized OS-ELM and CEOS-ELM as non-adaptive sequential ELM.

To the best of our knowledge, no previous single base ELM approach specifically addresses many concept drifts learning \cite{17}. However, some papers \cite{4,36} already discussed how the ELM implementation in adaptive environment.

Schaike, \textit{et.al.} \cite{36} proposed Online Pseudo Inverse Update Method (OPIUM). OPIUM is based on Greville's method as the incremental solutions to compute the pseudo inverse of matrix. The pseudo inverse computation can be solved incrementally as linear regression problems and can be adaptive which allows for non stationary data. The derivation of OPIUM is equivalent to the OS-ELM if the condition $c(k) \overset{\text{def}}{=} (I - H_{(k-1)}H_{(k-1)}^{-1})H_{(k)} = 0$ met at each iteration. This condition implies $H_{(k)}$ is a linear combination of the previous hidden layer $H_{(k-1)}$ and the simpler derivation of \cite{2} with right pseudo inverse become:

$$\hat{\beta} = TH^T = \overline{TH}^T(H^T H)^{-1}$$ (8)

Schaike, \textit{et.al.} defined $\psi$ as the cross correlation matrix between $T$ and $H$ ($TH^T$) and $\theta$ as the inverse of the auto correlation $H (H^T H)^{-1}$, so $\beta = \psi \theta$. According to Greville’s method, the solution for $\psi(k) = \psi(k-1) + T(k)H(k)$. And the solution for $\theta(k) = (H(k-1)H_{(k-1)}^T + H(k)H_{(k)}^T)^{-1}$, or in short writing as $\theta(k) = f(\theta(k-1), H(k))$.

Schaike, \textit{et.al.} proposed a simplified version named OPIUM light by computing only the on-diagonal element of $\theta(k)$. Schaike, \textit{et.al.} applied the OPIUM light for non-stationary data by using different weight $\alpha$ in determining $\beta_{(k)}$ for the most recent pair $(T(k), H(k))$ that appropriate for non-stationary mapping, which are:

$\psi(k) = (2 - \alpha)\psi(k-1) + \alpha T(k)H_{(k)}^T$, and for $\theta(k) = ((2 - \alpha)H_{(k-1)}H_{(k-1)}^T + \alpha H_{(k)}H_{(k)}^T)^{-1}$.

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 $\alpha$ as a non-stationary mapping parameter between input and output vectors.

Cao, \textit{et. al.} \cite{4} 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. Second, an uncertainty classifier based on ELM (UC-ELM) algorithm is 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 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.

2 Proposed Method

2.1 Theoritical Background of AOS-ELM

In sequential learning, some partial training data arrives in time sequential fashion: 
\{ (x_0, t_0), (x_1, t_1), \ldots, (x_k, t_k) \}. Learning is the process of constructing function \( \hat{\beta} \) to map between observation and its nature called (class) \cite{7}. When the number of training data \( N \to \infty \), we need to address the expected value of \( \beta(\infty) = \hat{\beta} \).

Learning from the data \( D_n \) is the process to select a function \( \beta_n \) from a class of \( B \) by minimizing of the empirical squared error 
\[ e_n(\beta) = \frac{1}{n} \sum_{i=1}^{n} (H_i \beta - T_i)^2 \]
with the error probability 
\[ L(\beta_n) = P\{ I_{\{H_i \beta_n \neq T_i \}} | D_n \} \] of the resulting classifier. According to \cite{7}, the empirical squared error minimization is consistent under general conditions.

**Theorem 2.1** Assume that \( B \) is a totally bounded class of functions. If \( \beta_n \in B \), then the classification rule obtained by minimizing the empirical squared error over \( B \) is strongly consistent, that is,

\[ P\{ \lim_{n \to \infty} L(\beta_n) = L^* \} \to 1 \]  \hspace{1cm} (9)

Based on Law of Large Numbers (LLN) theorem \cite{15} and Theorem 2.1 in sequential learning with the number of training data \( N \to \infty \), we can make sure the consistency of expected value of learning model is \( \hat{\beta} = H^T T \).

The Concept drift refers to an online supervised learning model when the relation between the input data and the target variable changes over time \cite{12}. If the learning model from Concept 1 \( \hat{\beta}_1 \in B_1 \) is bounded by hypothesis space \( \mathbb{R}^{m_1} \) and feature space \( \mathbb{R}^{d_1} \). And the learning model from Concept 2 \( \hat{\beta}_2 \in B_2 \) 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 \( \hat{\beta}_1 \) needs a transition space before it converges to the new learning model \( \hat{\beta}_2 \in B_2 \subset \mathbb{R}^{m_2} \). Our transition space idea was inspired by geometric approach for solving many problems in the fields of pattern recognition and machine learning \cite{6,34}.

For transition space, we propose two approaches: i) Assign the random coordinates in the new concept space. 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. The \((X_1)\) in 1-D coordinate has corresponding 2-D projection coordinates as \((X_1, 0)\)). 

Here, we relate the ELM theory to the context of AOS-ELM concept drift scenarios as follows:

1. Scenario 1: virtual drift (VD).

Huang, et. al. [17] explained interpolation theory from ELM point of view as stated by the following description:

**Theorem 2.2** Given any small positive value \(\epsilon > 0\), any activation function which is infinitely differentiable in any interval, and \(N\) arbitrary distinct samples \((x_i, t_i) \in \mathbb{R}^d \times \mathbb{R}^m\), there exists \(L < N\) such that for any input weight and bias pair \(\{a_i, b_i\}_{i=1}^L\) randomly generated from any interval of \(\mathbb{R}^d \times \mathbb{R}\), according to any continuous probability distribution, then with probability one, \(\|H\beta - T\| < \epsilon\). Furthermore, if \(L = N\), then with probability one, \(\|H\beta - T\| = 0\).

According to Theorem 2.2 and Learning Principle I of ELM Theory [18], the input weight and bias as hidden nodes \(H\) 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 \(\{a_i, b_i\}_{i=1}^L\) on any sequential stages and still make sure with probability one that \(\|H\beta - T\| < \epsilon\).

2. Scenario 2: real drift (RD).

Huang, et. al. [17] explained universal approximation capability of ELM as described by the following theorem:

**Theorem 2.3** Given any nonconstant piecewise continuous function \(g: \mathbb{R}^d \to \mathbb{R}\), if span \(\{g(a, b, x) : (a, b) \in \mathbb{R}^d \times \mathbb{R}\}\) is dense in \(L_2\), for any continuous target function \(f\) and any function sequence \(\{g(a_i, b_i, x)\}_{i=1}^L\) randomly generated according to any continuous sampling distribution, \(\lim_{L \to \infty} \|f - f_L\| = 0\) holds with probability one if the output weights \(\beta_i\) are determined by ordinary least square to minimize \(\|f(x) - \sum_{i=1}^L \beta_i g(a_i, b_i, x)\|\).

Based on Theorem 2.3 and inspired by the related works [3,26], 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 \(\beta_{(k-1)}\) has no knowledge about the new concept. ELM can approximate any complex decision boundary, as long as the output weights \(\beta_i\) are determined by ordinary least square to keep the minimum.

### 2.2 AOS-ELM Algorithms

In this section, we presented the AOS-ELM pseudo codes in the \(k^{th}\) sequential with \(X_{(k)}\) training input and \(T_{(k)}\) target to update Model\(_{(k)}\).

Basically, we have three pseudo codes, namely OSELMSeq (Algorithm \(\Box\)) as OS-ELM and CEOS-ELM pseudo codes; the AOSELMVDSeq (Algorithm \(\Box\) as AOS-ELM pseudo codes. The Matlab source code, data set and demo file implementation are available at https://github.com/abudiman250172/adaptive-OS-ELM.
codes tackling virtual drift; and AOSELMRDSeq (Algorithm 2 as AOS-ELM pseudo codes for addressing real drift. We can combine each pseudo code together to form a hybrid drift Algorithm. We can increase the hidden nodes using CEOS-ELM in Algorithm 3 after AOSELMVDSQ or AOSELMRDSeq. 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 Fig. 3) and marginalized the output result by defining the $Y_s$ class of concept $S$ is $s = \arg\max_y T(y_s)$. We scoped 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 that belong 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 $Y_s$ approximated to 1.

The systematic rules make AOS-ELM more flexible to handle complex consecutive drifts scenario. The AOS-ELM only stored the previous output weight $\beta_{L \times m}$ and auto correlation $K_{L \times L}$. The auto correlation $K$ 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 $Y$ class is $= \arg\max_y T(y)$. According to [22], the target values $\in \{0, 1\}$ is equivalent with $y \in \{-1, 1\}$.

**Algorithm 1** Algorithm OSELMSeq {OS-ELM Sequential }

```
Require: $X_{(k)} \in [-1, 1] R^{d \times N}, T_{(k)} \in [0, 1] R^{N \times m}, A_{(k)}, b_L, K_{(k-1)}, \beta_{(k-1)}$
Ensure: $\beta_{(k)}, K_{(k)}$
1: Compute $H_{(k)} = \g(A_{(k)} \cdot X_{(k)} + b_L)$
2: if $Increase\_Hidden\_Nodes == true$ then
3: $\Delta A_{d \times L} = \text{RandomNumbers}([-1, 1], R^{d \times L})$
4: $\Delta b_{L} = \text{RandomNumbers}([-1, 1], R^{l \times L})$
5: $A_{(k)} = [A_{(k)} \Delta A_{d \times L}]$
6: $b_L = [b_L \Delta b_{L}]$
7: Compute $\Delta H_{(k)} = \g(\Delta A_{d \times L} \cdot X_{(k)} + \Delta b_{L})$
8: Compute $K_{(k)} = f(K_{(k-1)}, H_{(k)}, \Delta H_{(k)})$
9: Compute $\beta_{(k)} = f(\beta_{(k-1)}, K_{(k)}, H_{(k)}, \Delta H_{(k)}, T_{(k)})$
{Using CEOS-ELM Method}
10: else
11: Compute $K_{(k)} = K_{(k-1)} + H_{(k)}^T H_{(k)}$
12: Compute $\beta_{(k)} = f(\beta_{(k-1)}, K_{(k)}, H_{(k)}, T_{(k)})$
13: end if
14: return $\beta_{(k)}, K_{(k)}$
```

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 becomes $S \times m$ output nodes.

3 Experiments

3.1 Experiments Design in Classification

To verify our method, we designed some experiments with the following purposes:

- To investigate the effectiveness of AOS-ELM on tackling three concept drift scenarios (VD, RD, HD) in two sequential patterns (sudden changes, recurring context). We used various data set starting with synthetic data set (SEA, STAGGER) then with real data set in handwritten recognition (MNIST, USPS). Each data set has different drift characteristics. This experiment is presented in Section 3.2 and 3.4. We also demonstrated the AOS-ELM capability as drift detection role in section 3.3 using SEA data set.

- To investigate the effectiveness of AOS-ELM on transfer learning to combine different data set sources. This experiment is presented in Section 3.4 using two data set sources (MNIST, and USPS) in handwritten recognition problem.

- To investigate the effect of hidden nodes increase in the drift events and how it impacts performance. This experiment is presented in section 3.5.

We used Matlab $^\text{TM}$ running on Microsoft Windows $^\text{TM}$ Computer with 4 cores 2.5 GHz processor and 8 GB memory.

Our experiments are organized as follows:

1. Simulation benchmark tests on the datasets that commonly used in concept drift handling of stream data, e.g. SEA [25] and STAGGER [23] (See Table 2a). Both datasets 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 that caused by discriminant function changes while the number of attributes and classes from all concepts are still same. The change type is sudden drift. The expected result is the classifier has good performance for the newest concept [24].

2. We tested our algorithm with real-world public data sets from MNIST numeric (0 to 9) [27] and the USPS alphanumeric (A to Z, 0 to 9) handwritten dataset [33].
We used original grey-level image attributes \([X_{grey}]\) of MNIST data set and the combination of \([X_{grey}]\) with additional attributes from the 9x9 bins histogram of orientated gradients \((X_{HOG})\) of grey-level image features \([29]\). For USPS, we added more data with Gaussian random and salt-pepper noises. Refer to Table 2a for detail data set information.

3. We designed the initial input weights and bias based on robust OS-ELM with regularization scalar \(c\) (ROS) \([16]\) and then based on initial random from the normal distribution (NORM). The activation function is sigmoid. The pseudo inverse function is the orthogonal projection using ridge regularization.

4. Let’s define the following concept as:
   - \(C_1\) is MNIST\([X_{grey}]\) class (1-6);
   - \(C_2\) is MNIST\([X_{grey}]\) class (7-10);
   - \(C_3\) is MNIST\([X_{grey}X_{HOG}]\) class (1-6);
   - \(C_4\) is MNIST\([X_{grey}X_{HOG}]\) class (7-10);
   - \(C_5\) is MNIST\([X_{grey}X_{HOG}]\) class (1-10);
   - \(C_6\) is USPS\([X_{grey}X_{HOG}]\) class (1-10, A-Z);

We followed the simulated concept drift methods in Dries, et.al \([10]\). We simulated sudden drift by splitting the composition into two groups, e.g., \(C_1\) and \(C_2\) and recurring context by shuffled the composition of \(C_1\) and \(C_2\). We set the sequential training flow to be the following drift equation:

We followed the simulated concept drift methods in Dries, et.al \([10]\). We simulated sudden drift by splitting the composition into two groups, e.g., \(C_1\) and \(C_2\) and recurring context by shuffled the composition of \(C_1\) and \(C_2\). We set the sequential training flow to be the following drift equation:

\begin{align*}
(a) \text{Experiment 1 - Virtual drift:} & \quad \text{MNIST}\([X_{grey}]\) \overset{VD}{\triangleright} \text{MNIST}\([X_{grey}X_{HOG}]\) \\
(b) \text{Experiment 2 - Real Drift:} & \quad \text{For recurring context: } \mathbf{C}_1 \overset{RD \text{ shuffled}}{\triangleright} \mathbf{C}_2 \\
& \quad \text{For sudden drift: } \mathbf{C}_1 \overset{RD \text{ shuffled}}{\triangleright} \mathbf{C}_2 \\
(c) \text{Experiment 3 - Hybrid Drift:} & \quad \mathbf{C}_1 \overset{RD \text{ shuffled}}{\triangleright} \mathbf{C}_2 \\
(d) \text{Experiment 4 - MNIST+USPS Transfer Learning:} & \quad \mathbf{C}_5 \overset{RD \text{ shuffled}}{\triangleright} \mathbf{C}_6
\end{align*}

5. We measured the performance based on Table 2c. 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 non-adaptive 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 \([24]\)). We also compared with adaptive ELM ensemble method (See Fig. 4). We designed the hierarchical ensemble using two models of ELM classifier with different roles (See Fig. 4). The first role is a binary classifier
Figure 4. Hierarchical ELM ensemble for MNIST+USPS Experiment. The gray shadow showed the new classifiers assembled when the new concept presented.

Table 1. Concept Drift Scenarios, Compared Methods and Sequential Patterns.
(a) The Experiment design scenarios

| Data Set          | VD | RD | HD | Compared Methods                  |
|-------------------|----|----|----|-----------------------------------|
| SEA               | -  | ✓  | -  | OS-ELM, CEOS-ELM, Kolter          |
| STAGGER           | -  | ✓  | -  | OS-ELM, CEOS-ELM, Kolter          |
| 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, Recurring Context | Additional attributes or classes    |
| USPS     | Recurring Context             | Additional attributes or classes    |

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 e.g. SEA concepts. The ensemble also applied outdated classifier pruning when the ensemble detects the previous attributes need to be replaced.

3.2 SEA and STAGGER Concepts Result

We addressed the question whether non-adaptive OS-ELM and CEOS-ELM with \(\delta L\) increase could handle the concept drift situation. We compared between AOS-ELM with no \(\delta L\) increase (AOS-ELM1) and with \(\delta L\) 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.
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 (10× trials)      | 60000 (×2)   | 10000 (×2) |
| USPS     | Holdout (10× 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 [23].            |
| Testing Accuracy                          | The accuracy measurement of the testing data set which 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 inter-rater agreement for categorical items.        |

The AOS-ELM has better accuracy with better recovery time (See Table 3a, 3b) than CEOS-ELM, whereas non-adaptive OS-ELM fails (See Fig. 5). The AOS-ELM2 improved the forgetting capability better than AOS-ELM1. In comparison with Kolter, et.al 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 [23], AOS-ELM outperformed DWM. (See Table 3a and 3b).

3.3 Concept Drift Detection

The drift detection works based on loss estimation (See Fig. 2) that compared current prediction accuracy with the previous feedback. Using similar method on [1][31], we can
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 $C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$.

| Method | Parameter | C1 | C2 | C3 | C4 |
|--------|-----------|----|----|----|----|
| OS-ELM | NORM      | 89.48± 0.33 | 86.47± 0.37 | 83.41± 0.93 | 86.32± 0.45 |
|        | ROS       | 89.51± 0.37 | 86.43± 0.38 | 83.49± 0.95 | 86.31± 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 | 54.71± 1.19 | 48.50± 10.10 | 90.43± 0.30 |
|        | ROS       | 65.78± 1.21 | 66.67± 1.19 | 64.09± 1.89 | 90.14± 1.34 |

(b) Testing accuracy in % for STAGGER with $C_1 \rightarrow C_2 \rightarrow C_3$.

| Method | Parameter | C1 | C2 | C3 |
|--------|-----------|----|----|----|
| 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± 5.66 | 70.91± 10.93 | 100.00± 0.00 |
|        | ROS       | 56.29± 9.56 | 69.41± 14.05 | 100.00± 0.00 |

evaluate the intersection point between accuracy decrease and increase in Fig. 8. If the consecutive loss performance exceeded a certain threshold, then drift warning status triggered. We measured the output performance from the new concept output and compared with the previous output. If met certain criteria, then the new AOS-ELM is committed. Otherwise, the previous AOS-ELM is rolled back.

3.4 MNIST and MNIST+USPS Result

We measured the testing accuracy based on holdout test data by 10× experiment trials. The results are as follows:

1. Experiment 1 - Virtual drift.

The AOS-ELM of $[X_{grey}, X_{HOG}]$ has Cohen’s kappa of testing accuracy 95.72 (0.21) % approximated to its non-adaptive ELM and offline ELM of $[X_{grey}, X_{HOG}]$ version with the same hidden nodes number $L = 2000$. It has better accuracy than single attribute $[X_{grey}]$ or $[X_{HOG}]$ only (See Table 4b). It proves our explanation in the theoretical background on Section 2.1.

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

2. Experiment 2 - Real drift

The final result as showed in Table 5, the AOS-ELM has better Cohen’s kappa performance for all concepts than ELM ensemble and little exceed to its non-adaptive and offline ELM. (Table 5b).
Figure 5. Predictive accuracy of AOS-ELM1 (black line △) and AOS-ELM2 (black dots △) compared with OS-ELM (blue line ×) and CEOS-ELM (blue dots ×).

Figure 6. Predictive accuracy of AOS-ELM in SEA for each concept with m output (See Fig. 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).

As well 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 8b).

3. Experiment 3 - Hybrid drift

The final result in Table 5c the AOS-ELM has better Cohen’s kappa performance for HD than ELM ensemble and approximate to its non-adaptive and offline ELM.
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 $10 \times$ trials.

(a) Benchmark result - non-adaptive OS-ELM and Offline ELM

| Performance | ELM Method | $[X_{grey}] (L=2000)$ | $[X_{HOG}] (L=200)$ | $[X_{grey}X_{HOG}] (L=2000)$ |
|-------------|------------|------------------------|----------------------|-----------------------------|
| 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$[X_{grey}] \overset{VD}{\rightarrow}$ MNIST$[X_{grey}X_{HOG}]$ | 96.15 ± 0.08 | 95.72 (0.21) |

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

4. 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 5d) and approximate to its non-adaptive and offline ELM. The AOS-ELM shows better recovery time than ELM ensemble in Fig. 7.

3.5 The effect of hidden nodes increase

The initial size of hidden nodes $L_0$ selection is important to have good generalization performance. Some studies [17, 21] suggested for hidden node size to be equal at least 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 and processing time, and probably not giving a significant result at the end. Thus, we have a requirement to increase $\delta L$ in sequential stage [26].

The experiment result in Table 6 shows the performance improved when certain hidden nodes size increase. We have different $L_0$ conditions: 2000, the rank of initial training data (666), and rank of total training data (713) and multiple conditions of $\delta L$.
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 10 × trials.

(a) Benchmark result - non-adaptive OS-ELM and Offline ELM

| Source            | Class   | Testing Accuracy | Cohen’s kappa |
|-------------------|---------|------------------|---------------|
|                   |         | OS-ELM           | Offline ELM   | OS-ELM         | Offline ELM |
| MNIST $[X_{grey}]$| (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 $[X_{grey}X_{HOG}]$| (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 $[X_{grey}X_{HOG}]$| (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) vs. AOS-ELM

| Source            | Concept | Testing Accuracy | Cohen’s kappa |
|-------------------|---------|------------------|---------------|
|                   |         | ELM ensemble     | AOS-ELM       | ELM ensemble   | AOS-ELM       |
| MNIST $[X_{grey}]$| $C_1$   | 94.58 ± 0.17     | 96.09 ± 0.12  | 93.54 ± 0.35   | 95.10 ± 0.31  |
|                   | $C_2$   | 91.60 ± 0.29     | 94.34 ± 0.16  | 89.94 ± 0.57   | 92.56 ± 0.48  |

(c) HD Experiment - ELM ensemble (3 classifiers, full memory, outdated classifier pruning) vs. AOS-ELM

| Source            | Concept | Testing Accuracy | Cohen’s kappa |
|-------------------|---------|------------------|---------------|
|                   |         | ELM ensemble     | AOS-ELM       | ELM ensemble   | AOS-ELM       |
| MNIST $[X_{grey}]$| $C_3$   | 94.48 ± 0.33     | 97.04 ± 0.18  | 93.42 ± 0.35   | 96.42 ± 0.26  |
| MNIST $[X_{grey}X_{HOG}]$| $C_4$ | 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, ) vs. AOS-ELM

| Source            | Concept | Testing Accuracy | Cohen’s kappa |
|-------------------|---------|------------------|---------------|
|                   |         | ELM ensemble     | AOS-ELM       | ELM ensemble   | AOS-ELM       |
| MNIST $[X_{grey}X_{HOG}]$ | $C_5$ | 88.17 ± 11.06    | 95.91 ± 0.12  | 86.94 ± 0.33   | 95.46 ± 0.22  |
| USPS $[X_{grey}X_{HOG}]$| $C_6$  | 99.80 ± 0.05     | 99.95 ± 0.03  | 99.79 ± 0.40   | 99.95 ± 0.02  |

on the drift event: No increase, 500, 1000 and 2000 using ROS parameters. However,
the larger $L_0$ has better influence than $\delta L$ increase.

1. 'Under-fitting' condition.

'Under-fitting' is the condition when the model does not fit the data well enough
that makes unconvergence. Based on an empirical experiment with $\delta L$ increase in
Table 6. Testing accuracy in Cohen’s kappa (kappa error) in % AOS-ELM MNIST for different $L_0$ and $\delta L$.

| 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   | 95.30 (0.26)  | 95.31 (0.22)    | **96.28 (0.20)**| 96.69 (0.20)    |
| RD       | 2000  | 94.71 (0.24)  | 94.93 (0.23)    | 95.39 (0.22)    | **95.42 (0.22)**|
|          | 630   | 91.30 (0.30)  | 91.67 (0.29)    | 93.61 (0.26)    | **93.64 (0.25)**|
|          | 713   | 91.71 (0.29)  | 92.70 (0.28)    | **93.82 (0.25)**| **94.29 (0.25)**|

Table 7. Predictive accuracy performance in AOS-ELM for MNIST using different parameters and $\text{Rank}(\hat{P})$ before and after $\delta L$ increase. Each experiment is repeated $100 \times$ trials to get the probability of predictive accuracy $\leq 50\%$.

| Scenario | $L_0$ | $\delta L$ | Batch Size | $c$ | Before | After | Pred. Acc. $\leq 50\%$ |
|----------|-------|------------|------------|-----|--------|-------|------------------------|
| RD       | 630   | 500        | 1000       | 10  | 630    | 1130  | 0%                     |
|          | 630   | 500        | 1000       | 10  | 630    | 640   | 0%                     |
|          | 630   | 500        | 1000       | 10  | 630    | 730   | 0%                     |
|          | 630   | 100        | 1000       | 10  | 630    | 730   | 0%                     |
|          | 630   | 100        | 500        | 10  | 730    | 730   | 0%                     |
|          | 630   | 100        | 100        | 10  | 730    | 730   | 0%                     |
|          | 630   | 100        | 100        | 10  | 730    | 730   | 0%                     |
|          | 630   | 100        | 100        | 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   | 2190  | 0%                     |
| VD       | 666   | 500        | 500        | 5   | 666    | 1116  | 0%                     |
|          | 666   | 500        | 500        | 5   | 666    | 1116  | 0%                     |
|          | 666   | 100        | 500        | 5   | 666    | 766   | 0%                     |
|          | 666   | 100        | 500        | 5   | 666    | 716   | 0%                     |
|          | 666   | 50         | 500        | 5   | 666    | 716   | 0%                     |

The sequential phase on Table 7, we investigated particular condition when the AOS-ELM classifier has a bad result. We realized the ELM performance is dependent upon finding general matrix inverse of $H$. Based on orthogonal projection method in CEOS-ELM, we can employ the rank value of $\hat{P}$ as evaluation parameter to detect ‘under-fitting’.

The $\hat{P}$ is approximation to matrix $(H^T H)^{-1}$. The full rank of $P_{L \times L}$ is ideally equal with $L$. However, certain condition in the sequential training, e.g. poor training data or poor learning parameter selection may cause the diagonal squared matrix $P$ has less diagonalizable [14], thus not full rank anymore.

In the sequential learning, we can compare the $\text{Rank}(\hat{P})$ 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 of ‘under-fitting’ condition to occur. The $\text{Rank}(\hat{P})$ 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 $\text{Rank}(\hat{P})$ as evaluation parameter is more efficient because we do not need to compute $\beta$.

2. Sudden drift.

On Table 8a and 8b, the hidden nodes increase can improve the forgetting capability on sudden drift (it reduced the accuracy of the outdated concept).
### Table 8. Average Testing accuracy in % for RD experiment. For SEA, we started from $C_2$ to $C_4$. For MNIST, we started from $C_1$ to $C_2$.

(a) Sudden drift effect caused by split training composition with hidden nodes $\delta L$ increase.

| Data  | AOS-ELM | $\delta L$ | Concept | Testing Accuracy  |
|-------|---------|------------|---------|-------------------|
| SEA   | $L_0 = 3000$ | 0          | $C_2$   | $90.00 \pm 0.59$  |
|       |         |            | $C_4$   | $90.24 \pm 0.61$  |
|       | 500     | 0          | $C_2$   | $66.29 \pm 1.12$  |
|       |         |            | $C_4$   | $90.12 \pm 0.52$  |
| MNIST | $L_0 = 2000$ | 0          | $C_1$   | $96.42 \pm 0.21$  |
|       |         |            | $C_2$   | $93.68 \pm 0.23$  |
|       | 500     | 0          | $C_1$   | $17.59 \pm 0.98$  |
|       |         |            | $C_2$   | $97.08 \pm 0.15$  |

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

| Data  | $\delta L$ | Composition | Concept | Testing Accuracy  |
|-------|------------|-------------|---------|-------------------|
| MNIST | 0          | Split       | $C_1$   | $96.42 \pm 0.21$  |
|       |            | Shuffled    | $C_2$   | $93.68 \pm 0.23$  |
|       | 500        | Split       | $C_1$   | $17.59 \pm 0.98$  |
|       |            | Shuffled    | $C_2$   | $97.08 \pm 0.15$  |
|       | 1000       | Split       | $C_1$   | $8.65 \pm 1.13$   |
|       |            | Shuffled    | $C_2$   | $97.64 \pm 0.18$  |

In CEOS-ELM, when $\delta L$ increase in the same time with drift, it makes

$$\begin{bmatrix} H_1 & 0 \\ H_2 & \Delta H_2 \end{bmatrix}$$

and the new concept target $$\begin{bmatrix} 0 \\ t_2 \end{bmatrix} \in T_2$$ in split composition, while previous concept $$\begin{bmatrix} t_1 \\ 0 \end{bmatrix} \in T_1$$. Thus, in the process of finding $\hat{\beta}$ become simplified because $[H_2\Delta H_2]$ is partially trained by $t_2$ only and not by $t_1$. Thus, it reduced the generalization capability of $[H_2\Delta H_2]$ to recognize $T_1$ problem.

### 4 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:

- $C_1$ is $sinc$ function with 50000 training/5000 testing;
- $C_2$ is $sinus$ function with 50000 training/5000 testing;
- $C_3$ is $gaussian$ function with 50000 training/5000 testing.

The sequential experiments are following drift equations:

1. Experiment 1 : $C_1 \xrightarrow{RD} C_2$
2. Experiment 2 : $C_1 \xrightarrow{RD} C_2 \xrightarrow{RD} C_3$

We presented the result on the following figures 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 that giving the best regression result of each concept. The AOS-ELM has $L_0 = 100$, $\delta L = 0$, and $sigmoid$ function. More drifts occurred will weaken the older concepts. Thus, it needs larger amplifier constant value.
5 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 \[37\].

In this experiment, we used KDD Cup 1999 Competition data set. The full dataset had 4898431 network packets and grouped to be 23 classes (One Normal class and 22
Figure 11. The changes on Service names and number of classes following the streaming data.

Table 9. Performance comparison of AOS-ELM and non-adaptive OS-ELM to validation data set C₂ in %.

(a) Benchmark result - non-adaptive OS-ELM

| Concept | Parameters | Testing Accuracy | Cohen’s kappa |
|---------|------------|------------------|---------------|
| C₁      | OS-ELM     | 50.87 ± 0.01     | 45.81 (0.54)  |
| C₂      | 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 |
|---------|------------|------------------|---------------|
| C₁ ≫ HD C₂ | AOS-ELM1     | 92.18 ± 2.73     | 91.38 (0.29)  |
| C₁ ≫ HD C₂ | AOS-ELM2     | 94.64 ± 0.06     | 94.10 (0.24)  |
| End of full C₂ | AOS-ELM1     | 93.45 ± 1.18     | 92.78 (0.27)  |
| End of full C₂ | AOS-ELM2     | 94.57 ± 0.11     | 94.02 (0.25)  |

attack names based on a signature-based detection) [11]. The dataset has a control information (CI) header for delivering the data in numerical and multi-categorical 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 the data set for the growing of service names and the number of class attack in the whole dataset on Fig. 11. The challenge in IDS dataset 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 Fig. 11), we designed one drift scenarios based on two concepts (Table 9b). C₁ has ten classes, and 37 service names, and C₂ has 23 classes and 70 service names. Total training data for each concept is 920000 packets. No data repetition from the previous event, except at the end of C₂ sequential training. The composition between C₁/C₂ on HD event is 230000/690000.

The validation data set of C₂ is selected from all packets from minority classes and randomly selected original majority classes (10422 packets). We used holdout method with 5× trials. We used AOS-ELM1 for δL = 0 and AOS-ELM2 for δL = 500 (Other ELM parameters are same: L₀ = 1000, NORM, sig). The AOS-ELM result in this experiment can approximate the non-adaptive OS-ELM on C₂ (See Table 9b).
6 Challenges and Future Research

- We need to investigate the optimum transition space that minimize the gap to the new concept learning model. In certain case, the AOS-ELM may have the ‘under-fitting’ condition and require larger training data to achieve the new convergence.

- We need to check the consistency of AOS-ELM for different pseudo-inverse methods (E.g., Greville’s method [36]).

We think some ideas for AOS-ELM future researches:

- The need for transfer learning to solve big data problem when the distribution data changes.

- The AOS-ELM integration with another ELM methods, e.g., Weighted OS-ELM for imbalanced learning [30], ELM Autoencoder (ELM-AE) [39], Stacked ELM [40], etc.

- A detail systematic explanation based on rule extraction [2] for AOS-ELM in handling adaptive environment.

7 Conclusion

The proposed method gives better adaptive capability than non adaptive OS-ELM and CEOS-ELM in term of retaining the recognition performance when handling concept drifts. It uses a simple line of code and easy to deploy especially for consecutive drifts, compared with adaptive ensemble methods. While most adaptive classifiers work differently for each 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 term of accuracy, simplicity, fast and flexible. However, in certain VD and HD cases, the AOS-ELM accuracy may not exceed the non adaptive sequential ELM, which include the future training data. In RD cases, the AOS-ELM has better accuracy. In a real data implementation, the non-adaptive 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 non-adaptive 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 that 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 the convergence to the expected learning model, we proposed the rank value of the pseudo inverse autocorrelation hidden nodes matrix as evaluation parameter to prevent ‘under-fitting’ 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, simpler and more resource efficient. However, the AOS-ELM does have some drawbacks. Any hidden node changes could impact all notions.
References

1. C. Alippi, G. Boracchi, and M. Roveri. Just-in-time classifiers for recurrent concepts. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(4):620–634, April 2013.

2. N. H. Barakat and A. P. Bradley. Rule extraction from support vector machines: A review. *Neurocomputing*, 74(1-3):178–190, 2010.

3. A. Budiman, M. Fanany, and C. Basaruddin. Constructive, robust and adaptive os-elm in human action recognition. In *Industrial Automation, Information and Communications Technology (IAICT), 2014 International Conference on*, pages 39–45, Aug 2014.

4. K. Cao, G. Wang, D. Han, J. Ning, and X. Zhang. Classification of uncertain data streams based on extreme learning machine. *Cognitive Computation*, pages 1–11, 2014.

5. G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16 – 28, 2014. 40th-year commemorative issue.

6. L. C. A. Corsten. Statistical methods: The geometric approach (david j. saville and graham r. wood). *SIAM Review*, 34(3):506–508, 1992.

7. L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

8. T. G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems, MCS ’00*, pages 1–15, London, UK, UK, 2000. Springer-Verlag.

9. P. Dongre and L. Malik. A review on real time data stream classification and adapting to various concept drift scenarios. In *Advance Computing Conference (IACC), 2014 IEEE International*, pages 533–537, Feb 2014.

10. A. Dries and U. Rückert. Adaptive concept drift detection. *Statistical Analysis and Data Mining*, 2(5-6):311–327, Dec. 2009.

11. J. Gama. Knowledge discovery from ubiquitous streams - datasets for concept drift.

12. J. a. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, Mar. 2014.

13. J. Gao, W. Fan, J. Han, and P. S. Yu. A general framework for mining concept-drifting data streams with skewed distributions. In *In Proc. SDM’07*, 2007.

14. G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.).* Johns Hopkins University Press, Baltimore, MD, USA, 1996.

15. C. M. Grinstead and J. L. Snell. *Introduction to Probability*. AMS, 2003.

16. 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*, pages 1077–1086, Berlin, Heidelberg, 2007. Springer-Verlag.
17. G. Huang, G.-B. Huang, S. Song, and K. You. Trends in extreme learning machines: A review. *Neural Networks*, 61(0):32 – 48, 2015.

18. G.-B. Huang. An insight into extreme learning machines: Random neurons, random features and kernels. *Cognitive Computation*, 6(3), 2014.

19. G.-B. Huang. What are extreme learning machines? filling the gap between frank rosenblatt’s dream and john von neumann’s puzzle. *Cognitive Computation*, 7:263–278, June 2015.

20. 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*, 42(2):513–529, 2012.

21. G.-B. Huang, Q. Y. Zhu, and C. K. Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

22. A. Iosifidis. Extreme learning machine based supervised subspace learning. *Neurocomputing*, 167:158 – 164, 2015.

23. J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *J. Mach. Learn. Res.*, 8:2755–2790, Dec. 2007.

24. L. Kuncheva. Classifier ensembles for changing environments. In *Multiple Classifier Systems*, volume 3077 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg, 2004.

25. L. I. Kuncheva. Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. In *2nd Workshop SUEMA 2008 (ECAI 2008)*, pages 5–10, 2008.

26. Y. Lan, Y. C. Soh, and G.-B. Huang. A constructive enhancement for online sequential extreme learning machine. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 1708–1713, June 2009.

27. Y. LeCun and C. Cortes. MNIST handwritten digit database, 2010.

28. N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan. A fast and accurate online sequential learning algorithm for feedforward networks. *Neural Networks, IEEE Transactions on*, 17(6):1411–1423, Nov 2006.

29. O. Ludwig, D. Delgado, V. Goncalves, and U. Nunes. Trainable classifier-fusion schemes: An application to pedestrian detection. In *Intelligent Transportation Systems, 2009. ITSC ’09. 12th International IEEE Conference on*, pages 1–6, Oct 2009.

30. B. Mirza, Z. Lin, and K.-A. Toh. Weighted online sequential extreme learning machine for class imbalance learning. *Neural Processing Letters*, 38(3):465–486, 2013.

31. K. Nishida and K. Yamauchi. Adaptive classifiers-ensemble system for tracking concept drift. In *Machine Learning and Cybernetics, 2007 International Conference on*, volume 6, pages 3607–3612, Aug 2007.

32. S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

33. S. Roweis. Data for matlab hackers handwritten digits.
34. R. Strack. *Geometric Approach to Support Vector Machines Learning for Large Datasets*. PhD thesis, Richmond, VA, USA, 2013. AAI3563551.

35. W. N. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-scale classification. pages 377–382. ACM, 2001.

36. A. van Schaik and J. Tapson. Online and adaptive pseudoinverse solutions for \{ELM\} weights. *Neurocomputing*, 149, Part A:233 – 238, 2015.

37. S. K. Wagh, V. K. Pachghare, and S. R. Kolhe. Article: Survey on intrusion detection system using machine learning techniques. *International Journal of Computer Applications*, 78(16):30–37, September 2013.

38. Q. Yang. Transfer learning beyond text classification. In Proceedings of the 1st Asian Conference on Machine Learning: Advances in Machine Learning, ACML ’09, pages 10–22, Berlin, Heidelberg, 2009. Springer-Verlag.

39. N. Zhang, S. Ding, and Z. Shi. Denoising laplacian multi-layer extreme learning machine. *Neurocomputing*, 171:1066 – 1074, 2016.

40. H. Zhou, G.-B. Huang, Z. Lin, H. Wang, and Y. C. Soh. Stacked extreme learning machines. *Cybernetics, IEEE Transactions on*, 45(9):2013–2025, 2015.

41. I. Zliobaite. Learning under Concept Drift: an Overview. *Computing Research Repository*, abs/1010.4, 2010.