IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 26, 2019, accepted December 8, 2019, date of current version December 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2959032

Online Sequential Extreme Learning Machine With Dynamic Forgetting Factor

WEIPENG CAO[1], ZHONG MING[1], ZHIWU XU[1], JIYONG ZHANG[2], AND QIANG WANG[3]
[1]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
[2]School of Automation, Hangzhou Dianzi University, Hangzhou 310018, China
[3]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China

Corresponding author: Qiang Wang (wenjunwang.nudt@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672358 and Grant 61836005, and in part by the Guangdong Science and Technology Department under Grant 2018B010107004.

ABSTRACT Online sequential extreme learning machine (OS-ELM) and its variants provide a promising way to solve data stream problems, but most of them do not take the timeliness of the problems into account, which may degrade the performance of the model. The main reason is that these algorithms are unable to adapt to the latest data accordingly when the distribution of the data stream changes. To mitigate this limitation, the forgetting factor is introduced into the relevant models, which is used to balance the relative importance of past data and new data when necessary. However, there is no efficient way to set the forgetting factor properly so far. In this paper, we have developed a novel updating strategy for setting the forgetting factor and proposed a dynamic forgetting factor based OS-ELM algorithm (DOS-ELM). In the sequential learning phase of DOS-ELM, the forgetting factor can be adjusted dynamically according to the change degree of the model accuracy in each learning epoch. This updating process does not require setting any parameters artificially and thus greatly improves the flexibility of the model. The experimental results on ten classification problems, five regression problems, one time-series problem show that DOS-ELM can deal well with both stationary and non-stationary data stream problems. In addition, we have extended DOS-ELM to an online deep model named ML-DOS-ELM, which can handle more complex tasks such as the face recognition problem and the handwritten digit recognition problem. Our experimental evaluations show that both DOS-ELM and ML-DOS-ELM can achieve higher prediction accuracy compared to the other similar algorithms.

INDEX TERMS Extreme learning machine, online sequential learning, forgetting factor, timeliness.

I. INTRODUCTION

Recently artificial neural networks have achieved significant breakthroughs in many fields, such as speech recognition [1], action recognition [49], image processing [2], [3], and natural language processing [4]. Most of these algorithms use the batch learning mode [38], which is characterized by the fact that whenever the new data is received the model will be retrained together with the past data. Such a training mechanism makes it difficult for batch learning algorithms to deal with the online learning problems (i.e., data stream learning problems) such as the short-term energy consumption prediction problem. The main reason is that with the continuous arrival of new data, the batch learning based training mechanism becomes more time-consuming, and demands a higher usage of memory.

To alleviate the above problem, online sequential learning algorithms are proposed, which do not require any prior knowledge as to the number of training samples that will be presented, only the newly arrived samples are seen and learned. Therefore, the memory usage of these algorithms is relatively smaller than the batch learning algorithms. Some notable neural networks-based online sequential learning techniques include the back-propagation based algorithms (e.g., SGBP [5]), neural networks with radial basis function nodes (e.g., GAP-RBF [6]), extreme learning machine (ELM) based algorithms (e.g., OS-ELM [7] and Fuzziness based OS-ELM [8]). The first two algorithms (i.e., SGBP and GAP-RBF) adopt the iterative learning mechanism based on the error back propagation. They can handle the new data

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Tong.

one by one but always suffer from long training time. While ELM based algorithms (e.g., OS-ELM) are able to learn the new data one by one and chunk by chunk with fixed or varying size. In addition, these algorithms can achieve much faster learning speed than others in some cases due to their non-iterative learning mechanism [9]. Furthermore, only the number of hidden layer nodes needs to be specified in the ELM based algorithms, whereas there are many hyper-parameters (e.g., learning rate, momentum, etc.) that to be specified and fine-tuned in the others. OS-ELM and its variants have been applied to many fields, such as class imbalance learning [10], [43], [48], indoor localization [11], engine air-ratio control [12], and gas utilization ratio prediction [47].

The aforementioned algorithms assume that the distribution of data stream is stationary. In fact, the distribution may change with time in some practical scenarios such as the stock data and meteorological data. For these scenarios, one needs to consider the timeliness of the problems and give higher weights to the new data when necessary, in this way the online model can be better adapted to the new environment. However, detecting and coping with the changes in the distribution of a data stream is a very challenging task [13], [14].

To deal with the above issue, several OS-ELM based online sequential learning algorithms have been proposed such as FOS-ELM [15], OS-ELMK [16], FOK-ELM [17], TOS-ELM [18], WOS-ELM [19], and some other advanced variants [39]–[45]. In this paper, we focus on the algorithms with a controlling parameter named forgetting factor such as TOS-ELM and WOS-ELM, which can balance the relative importance of new data and past data and adjust the model to pay more attention to the new data when the concept drift is detected. However, these algorithms do not give a convenient and efficient way to set the value of forgetting factor. Take WOS-ELM as an example, the updating function of its forgetting factor contains two predefined parameters (i.e., the setpoint error and the minimum of the forgetting factor), which are hard to be set properly by empirical methods. In addition, there is only one hidden layer in their network structure, which may make them unable to extract high-level features from some complex tasks such as the image processing problem.

To solve the above problem, we developed a novel updating strategy for the forgetting factor and proposed an OS-ELM with dynamic forgetting factor algorithm (DOS-ELM) in this paper. In DOS-ELM, the change degree of the model accuracy in each learning epoch is used as the signal for adjusting the forgetting factor. Specifically, if the prediction accuracy of the model updated with the new data is lower than that of the previous model, the forgetting factor will be reduced according to a predefined rule, which means that the weight of the past data is reduced while the weight of the new data is relatively enhanced, and vice versa. Notice that there is no parameter to be set artificially in the updating process of the forgetting factor, which implies that the proposed method can avoid the interference of human factors on the performance of the model. Experimental results on ten classification problems (including four data stream problems with clear non-stationary properties), five regression problems, and one time series problems show that the proposed DOS-ELM has better generalization performance than OS-ELM [7], TOS-ELM [18], and WOS-ELM [19].

In addition, inspired by the idea of OS-ELM-AE [20], we have extended DOS-ELM to an online deep model named Multiple hidden Layers DOS-ELM (ML-DOS-ELM). In the network structure of ML-DOS-ELM, there are at least two hidden layers. The experimental results on a complex face recognition problem and a popular handwritten digit recognition problem show that ML-DOS-ELM can achieve higher prediction accuracy than OS-ELM, DOS-ELM, and Multi-layer-OS-ELM [20].

The main contributions of this paper are as follows.

1) A novel online sequential learning algorithm with dynamic forgetting factor named DOS-ELM is proposed, which provides a new updating strategy for the forgetting factor. The proposed updating strategy does not require setting any parameters artificially and thus avoids the interference of human factors on the model performance.
2) We extend DOS-ELM to a multi-layer online model named ML-DOS-ELM, which can extract more high-level features from the training data and provide a new way to deal with complex problems in online learning scenarios.
3) The proposed DOS-ELM provides a unified framework for dealing with classification, regression, time series, and image processing tasks. In addition, we have performed thorough experimental evaluations, which justify that the proposed algorithm can deal well with the data stream problems with clear non-stationary properties and stationary properties.

The remainder of this paper is organized as follows. Section II briefly reviews ELM and OS-ELM algorithms. The proposed DOS-ELM algorithm is given in Section III. The details of the experimental results and the corresponding analysis are described in Section IV. Section V provides conclusions and future works.

II. REVIEW OF ELM AND OS-ELM

Extreme learning machine (ELM) is a special type of feed-forward neural network, in which the input weights and hidden bias are assigned randomly and then remain the same throughout the training process while the output weights are obtained analytically. This non-iterative learning mechanism enables it to achieve much faster training speed than the traditional neural networks in many scenarios [50], [51]. This section briefly reviews the training mechanism of ELM and online sequential ELM (OS-ELM).

A. EXTREME LEARNING MACHINE (ELM)

Extreme Learning Machine (ELM), first proposed by Huang G.B. in 2004 [21], has the same training mechanism as several

early neural networks with random weights (NNRW) such as RVFL [22] and FNNRW [23]. In recent years, ELM has attracted many attentions and lots of ELM-based algorithms have been proposed and applied to many fields given its strengths in model training [24]–[27].

A typical network structure of ELM is shown in FIGURE 1, where ω denotes the input weights (i.e., the weights between the input layer and hidden layer) and b refers to the hidden bias (i.e., the thresholds of the hidden layer nodes), both of them are assigned randomly; β denotes the output weights (i.e., the weights between the hidden layer and output layer), which are obtained analytically. In addition, d, L, and m are the number of input layer nodes, the number of hidden layer nodes, and the number of output layer nodes, respectively. o denotes the predicted results of the model.

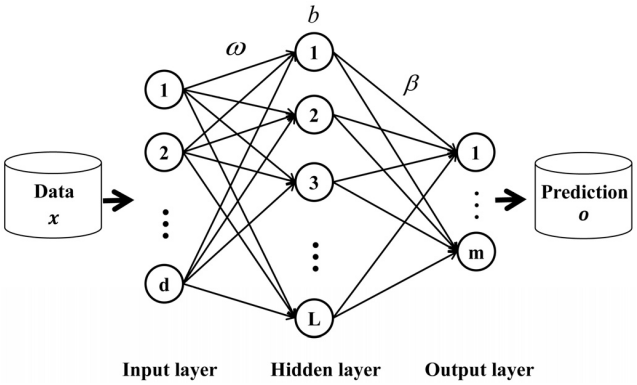


FIGURE 1. The network structure of ELM with a single hidden layer.

Given a dataset $D = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m\}_{i=1}^N$, an ELM with L hidden layer nodes and an activation function $G(\cdot)$ can be modeled as:

$$\sum_{i=1}^{L} \beta_i G(\omega_{ij} \cdot x_j + b_i) = t_j, \quad j = 1, 2, \ldots, N \tag{1}$$

which also can be rewritten compactly as

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

where

$$H = \begin{pmatrix} G(\omega_1 \cdot x_1 + b_1) & \cdots & G(\omega_L \cdot x_1 + b_L) \\ \vdots & \ddots & \vdots \\ G(\omega_1 \cdot x_N + b_1) & \cdots & G(\omega_L \cdot x_N + b_L) \end{pmatrix}_{N \times L},$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}, \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m}$$

The optimization object of ELM is as follows:

$$\min_{\|\beta\|} \left(\min \sum_{i=1}^{N} \|t_i - o_i\|^2 \right) \tag{3}$$

where o_i is the predicted label of ELM model for the i_{th} sample, t_i is the real label of the i_{th} sample, and N is the number of samples.

According to the ELM theory, we can solve this problem by using the Moore–Penrose generalized inverse method:

When $N > L$,

$$\beta = H^+ T = (\mathrm{H}^T \mathrm{H})^{-1} \mathrm{H}^T \mathrm{T} \tag{4}$$

When $N < L$,

$$\beta = H^+ T = \mathrm{H}^T (\mathrm{H}\mathrm{H}^T)^{-1} \mathrm{T} \tag{5}$$

When $N = L$,

$$\beta = H^{-1} T \tag{6}$$

B. ONLINE SEQUENTIAL EXTREME LEARNING MACHINE (OS-ELM)

Online Sequential Extreme Learning Machine (OS-ELM) [7] is an ELM based online sequential learning algorithm, which combines the advantages of ELM (e.g., fast learning) and the updating characteristics of online learning strategy (e.g., only the newly arrived data are seen and learned) and shows great potential in dealing with the data stream problems.

The training process of OS-ELM includes two phases: the initialization phase and the online sequential learning phase. Given a training dataset $D = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m\}_{i=1}^N$, the learning process of OS-ELM can be summarized as follows.

1) INITIALIZATION PHASE

Given an initial training dataset $D_0 = \{(x_j, t_j|x_j \in R^d, t_j = R^m)\}_{j=1}^{N_0}$ from D, the initial ELM model can be modeled and trained based on (1)-(6). The output matrix of the hidden layer and the output weights of the initial ELM model are marked as H_0 and β_0, respectively. Set $k = 0$ and $P_0 = (H_0^T H_0)^{-1}$.

2) ONLINE SEQUENTIAL LEARNING PHASE

1) Update the output matrix of the hidden layer as $H_{k+1} = [H_k^T, H_{k+1}^T]^T$ when the $(k+1)_{th}$ chunk of new dataset D_{k+1} is given. Here, H_{k+1} denotes the corresponding random feature mapping matrix (i.e., the current output matrix of the hidden layer) of the new dataset.
2) Update the current output weights by using $\beta_{k+1} = \beta_k + P_{k+1} H_{k+1}^T (T_{k+1} - H_{k+1}\beta_k)$, in which $P_{k+1} = P_k - P_k H_{k+1}^T (I + H_{k+1} P H_{k+1}^T)^{-1} H_{k+1} P_k$ and T_{k+1} denotes the labels of the new dataset.
3) Set $k = k + 1$. Go back to step 1) until all the new data are learned.

III. ONLINE SEQUENTIAL EXTREME LEARNING MACHINE WITH DYNAMIC FORGETTING FACTOR

In the online sequential learning phase of OS-ELM, when the new data arrives, the model is updated according to the following optimization object:

$$\min \left\| \begin{bmatrix} H_i \\ H_{i+1} \end{bmatrix} \beta - \begin{bmatrix} T_i \\ T_{i+1} \end{bmatrix} \right\| \tag{7}$$

From (7), it can be observed that the past data and new data have the same weight coefficients in OS-ELM (i.e., both are 1), which means that OS-ELM regards their importance as the same. In other words, OS-ELM does not consider the timeliness of the training data. Therefore, we can infer that this method can deal with data stream problems with stable distribution.

However, in some real-life industrial application scenarios, the distribution of the data stream may gradually change due to the equipment wear or other reasons. In this case, the model is required to be able to discover the changes of the data stream as soon as possible and adjust the relative importance of new data and past data to better adapt to the new environment. In this case, the vanilla OS-ELM may not work very well.

To solve the above problem, several improved algorithms have been proposed in recent years such as FOS-ELM [15], OS-ELMK [16], FOK-ELM [17], TOS-ELM [18], and WOS-ELM [19]. One common feature of these algorithms is that they can balance the relative importance of past data and new data by a control parameter named forgetting factor, which can be regarded as the consideration of the model to the timeliness of data stream problems. However, there is no convenient mechanism to set a proper value for the forgetting factor in these algorithms. In addition, the network structures of these algorithms only have one hidden layer, which may limit their applications in some complex tasks such as image processing problems.

To deal with the above issues, we propose a novel online sequential learning algorithm with dynamic forgetting factor named DOS-ELM in this paper. DOS-ELM shows three advantages as follows.

1) The forgetting factor of DOS-ELM can be automatically and dynamically adjusted according to the iterative error, avoiding the instability of the model caused by artificial hyperparameters such as the setpoint error in the WOS-ELM;
2) DOS-ELM can be easily extended into a deep online model with multiple hidden layers (ML-DOS-ELM), which can be used to deal with some complex tasks in the online learning scenario;
3) DOS-ELM provides a unified framework for dealing with classification, regression, and time-series problems.

The details of DOS-ELM and ML-DOS-ELM are given as follows.

A. DOS-ELM

Different from OS-ELM, in the online sequential learning phase of DOS-ELM, the optimization object becomes

$$\min \left\| \begin{bmatrix} \lambda H_0 \\ H_1 \end{bmatrix} \beta - \begin{bmatrix} \lambda T_0 \\ T_1 \end{bmatrix} \right\| \tag{8}$$

where λ is the forgetting factor and $\lambda \in [0,\ 1]$. $\lambda = 1$ implies that the importance of the new data and past data is same, which is the case of OS-ELM. $\lambda < 1$ implies that the importance of the past data is lower than that of the new data.

From (8), we can deduce that

$$\beta^{(1)} = K_1^{-1} \begin{bmatrix} \lambda H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} \lambda T_0 \\ T_1 \end{bmatrix} \tag{9}$$

where $K_1 = \begin{bmatrix} \lambda H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} \lambda H_0 \\ H_1 \end{bmatrix}$. K_1 can also be rewritten as

$$K_1 = [\lambda H_0\ H_1] \begin{bmatrix} \lambda H_0 \\ H_1 \end{bmatrix} = \lambda^2 K_0 + H_1^T H_1 \tag{10}$$

where $K_0 = H_0^T H_0$.

And

$$\begin{aligned} \begin{bmatrix} \lambda H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} \lambda T_0 \\ T_1 \end{bmatrix} &= \lambda^2 H_0^T T_0 + H_1^T T_1 \\ &= \lambda^2 K_0 K_0^{-1} H_0^T T_0 + H_1^T T_1 \\ &= \lambda^2 K_0 \beta^{(0)} + H_1^T T_1 \\ &= (K_1 - H_1^T H_1)\beta^{(0)} + H_1^T T_1 \\ &= K_1 \beta^{(0)} - H_1^T H_1 \beta^{(0)} + H_1^T T_1 \end{aligned} \tag{11}$$

Combining (9) and (11), we can obtain that

$$\begin{aligned} \beta^{(1)} &= K_1^{-1} \begin{bmatrix} \lambda H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} \lambda T_0 \\ T_1 \end{bmatrix} \\ &= K_1^{-1}(K_1 \beta^{(0)} - H_1^T H_1 \beta^{(0)} + H_1^T T_1) \\ &= \beta^{(0)} + K_1^{-1} H_1^T (T_1 - H_1 \beta^{(0)}) \end{aligned} \tag{12}$$

When the $(k+1)_{th}$ chunk of the new data arrives, we can obtain the following generalized formula by generalizing the coefficients of (10) and (12).

$$\begin{aligned} K_{k+1} &= \lambda^2 K_k + H_{k+1}^T H_{k+1} \\ \beta^{(k+1)} &= \beta^{(k)} + K_{k+1}^{-1} H_{k+1}^T (T_{k+1} - H_{k+1} \beta^{(k)}) \end{aligned} \tag{13}$$

where K_{k+1}^{-1} is updated by using the Woodbury formula [28]

$$\begin{aligned} {K_{k+1}}^{-1} &= (\lambda^2 K_k + H_{k+1}^T H_{k+1})^{-1} \\ &= (\lambda^2 K_k)^{-1} \\ &\quad - (\lambda^2 K_k)^{-1} H_{k+1}^T [I + H_{k+1} (\lambda^2 K_k)^{-1} H_{k+1}^T]^{-1} \\ &\quad \times H_{k+1} (\lambda^2 K_k)^{-1} \\ &= \lambda^{-2} K_k^{-1} \\ &\quad - \lambda^{-4} K_k^{-1} H_{k+1}^T (I + \lambda^{-2} H_{k+1} K_k^{-1} H_{k+1}^T)^{-1} \\ &\quad \times H_{k+1} K_k^{-1} \end{aligned} \tag{14}$$

Let $P_{k+1} = K_{k+1}^{-1}$, the model updating equations can be rewritten as

$$\begin{aligned} P_{k+1} &= \lambda^{-2} P_k - \lambda^{-4} P_k H_{k+1}^T (I + \lambda^{-2} H_{k+1} P_k H_{k+1}^T)^{-1} \\ &\quad \times H_{k+1} P_k \\ \beta^{(k+1)} &= \beta^{(k)} + P_{k+1} H_{k+1}^T (T_{k+1} - H_{k+1} \beta^{(k)}) \end{aligned} \tag{15}$$

It is noted that DOS-ELM degrades into the OS-ELM when $\lambda = 1$. The forgetting factor λ is updated by

$$\begin{aligned} &\lambda = \lambda - \frac{1}{5\pi} atan(E) \\ &s.t.\ if\ \lambda > 1 \quad then\ \lambda = 1 \end{aligned} \tag{16}$$

where E denotes the difference between the accuracy of the current model that has learned the latest data and that of the previous model that does not studied the latest data, $atan(\cdot)$ is the arctangent function.

Here we give the details of calculating the accuracy of the current model. When the new data chunk arrives (denoted as D_{k+1}), DOS-ELM updates the output weights using the formula derived in this paper and obtains the current model. Then we use the model to predict the labels of all samples in D_{k+1}. The accuracy of the current model can be obtained based on the difference between the predicted labels of the samples and the real labels of the samples. Then, let it subtract the accuracy of the previous model on D_k to obtain the change degree E of the model accuracy in each learning epoch. Later, the value of the forgetting factor will be dynamically adjusted by substituting E into the updating formula of the forgetting factor.

Besides, we give the reason for choosing arctangent function as the adjustment function of forgetting factor. Our algorithm is mainly used to solve the gradual concept that may appear in the data stream. Compared with other types of concept drift, the change of data stream is relatively stable in the case of gradual concept. Considering this characteristic, we choose the arctangent function (denoted as $atan(\cdot)$) to calculate the value of the forgetting factor according to the change degree (denoted as E) of the model accuracy in each learning epoch, because the arctangent function has good smoothness. The output range of $atan(E)$ is $(-\frac{\pi}{2}, \frac{\pi}{2})$. When $|E| > 15$, the value of $atan(E)$ tends to be saturated. If we choose the hyperbolic tangent function (denoted as $tanhx(E)$) as the activation function, when $|E| > 3$, the value of $tanhx(E)$ tends to be saturated. In other words, the mapping interval of the hyperbolic tangent function is more compact. Therefore, the $atan(\cdot)$ can adjust the value of the forgetting factor in a wider range than that of the $tanhx(\cdot)$.

The $\frac{1}{5\pi}$ in (16) is actually the zoom coefficient of $atan(E)$, which is used to scale the output range of $atan(E)$ to $(-0.1,\ 0.1)$ In this way, the change of the forgetting factor can be made more stable, thus avoiding the performance fluctuation of the model caused by the excessive change of the forgetting factor [37]. The value of $\frac{1}{5\pi}$ is determined by multiple experiments. In the experiment, we chose $\frac{1}{2\pi}$, $\frac{1}{3\pi}$, $\frac{1}{4\pi}$, $\frac{1}{5\pi}$, $\frac{1}{6\pi}$, $\frac{1}{7\pi}$, and $\frac{1}{8\pi}$ as the scaling coefficient respectively. Through a large number of experiments, we found that $\frac{1}{5\pi}$ as the scaling coefficient can make the model have the highest prediction accuracy on all datasets in the paper. In other words, setting the value of the scaling coefficient to $\frac{1}{5\pi}$ can work well regardless of the dataset type. Therefore, the only value that dynamically changes in (16) is E (i.e., the change degree of the model accuracy in each learning epoch), and the value of E is automatically calculated during the process of sequential learning. So, we say no parameter needs to be set artificially in the updating process of the forgetting factor in our method. Compared to other algorithms such as WOS-ELM, our method can avoid the interference of human factors on the performance of the model. It is noted that any bounded monotonic function can be used as the adjustment function for the forgetting factor.

It can be inferred from (16) that when $E > 0$, that is, the model accuracy degrades after learning the new data, the forgetting factor λ will gradually decrease with the increase of E; when $E = 0$, λ remains unchanged; when $E < 0$, that is, the model accuracy is improved after learning the new data, the forgetting factor λ will gradually increase until $\lambda = 1$.

The proposed DOS-ELM algorithm can be summarized as follows.

Algorithm 1 DOS-ELM Algorithm

Input: A training dataset $D = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m\}_{i=1}^N$, the activation function $G(\cdot)$, and the number of hidden layer nodes L. Set the counter $k = 0$ and the initial forgetting factor $\lambda = 1$.

Output: The output weights of DOS-ELM, marked as β.

Initialization phase:

Initialize the learning process using the same method as OS-ELM. As to the initial model, H_0 denotes the output matrix of the hidden layer, and β_0 denotes the output weights. Let the intermediate variable $P_0 = (H_0^T H_0)^{-1}$. Calculate the prediction accuracy of the initial model and mark it as ACC_0.

Online sequential learning phase:

1) When the $(k+1)_{th}$ chunk of the new data D_{k+1} arrive, update the hidden layer output matrix as $H_{k+1} = [\lambda H_k^T,\ H_{k+1}^T]^T$.
2) Update the output weights β_{k+1} and the intermediate variable P_{k+1} according to (15).
3) Calculate the prediction accuracy of the current model and mark it as ACC_{k+1}. Then calculate the accuracy change of the model by $E = ACC_{k+1} - ACC_k$ and use it to update the forgetting factor λ through (16).
4) Let $k = k + 1$ and go to the step 1) until all the new data are learned.

B. ML-DOS-ELM

One reason for the success of deep learning is that the model usually contains more than one hidden layer, which can make the model able to extract more high-level features from the training data. In this sense, we extend DOS-ELM to a deep online model that contains multiple hidden layers to deal with some complex tasks.

Mirza *et al.* [20] have proposed an OS-ELM based autoencoder named OS-ELM-AE to extract high-level features from the training data. In the network structure of OS-ELM-AE, the output is set to be equal to the input and the optimization object is to minimize the reconstruction error. It is noted that the input weights and hidden biases need to be orthogonalized in OS-ELM-AE. The training process of OS-ELM-AE is the same as that of the OS-ELM, which can be summarized as follows.

Algorithm 2 OS-ELM-AE Algorithm

Initialization phase:

Given an initial training dataset $D_0 = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m\}_{i=1}^{N_0}$, the output weighs $\beta^{(0)}$ can be obtained by

$$\beta^{(0)} = P_0 H_0^T X_0 \tag{17}$$

where $P_0 = (H_0^T H_0)^{-1}$ and $X_0 = [x_1, x_2, \cdots, x_{N_0}]$.

Online sequential learning phase:

When the $(k+1)_{th}$ chunk of new dataset D_{k+1} is given, update the output weight $\beta^{(k+1)}$ and the intermediate variable P_{k+1} by

$$\begin{aligned} P_{k+1} &= P_k - P_k H_{k+1}^T (I + H_{k+1} P H_{k+1}^T)^{-1} H_{k+1} P_k \\ \beta^{(k+1)} &= \beta^{(k)} + P_{k+1} H_{k+1}^T (X_{k+1} - H_{k+1} \beta^{(k)}) \end{aligned} \tag{18}$$

Inspired by the above work, we use OS-ELM-AE to extract high-level features from the training data and combine them with DOS-ELM to build a deep online model named ML-DOS-ELM. In ML-DOS-ELM, the dynamic adjustment strategy of forgetting factor in DOS-ELM is inherited, and the multi-hidden layers network structure enables it to extract more high-level features from the training data.

The details of ML-DOS-ELM algorithm can be summarized as follows.

Remark: In general, the output features of the hidden layers of ML-DOS-ELM are extracted by using OS-ELM-AEs, and the output weights between the last hidden layer and the output layer are analytically obtained. In other words, the parameters of the hidden layers of ML-DOS-ELM are initialized with OS-ELM-AEs, which perform layer-wise unsupervised learning. Specifically, each OS-ELM-AE is an auto-encoder with the same input and output. It is noted that the input weights and hidden biases of OS-ELM-AE should be orthogonalized, which has been proven helpful for the model to extract and transform features [35]. The output weights of OS-ELM-AE are used to learn the transformation from the feature space to the input data. The output of the hidden layer of each previous OS-ELM-AE unit acts as the input of the next OS-ELM-AE, which enables the model to process and extract data features at multiple levels. We can use OS-ELM-AEs to extract and stack the features from the training data to form a deep architecture by using similar methods like stacked auto-encoder [36]. In this way, we can achieve multiple conversion and extraction of data features. Then we can deduce the updating formulas of the parameters between different data chunks (see (18) for details) and add the proposed dynamic updating strategy of the forgetting factor to the formulas. When new data arrives, ML-DOS-ELM will dynamically adjust the value of the forgetting factor according to the change degree of the model accuracy in each learning epoch. Compared with the single hidden layer DOS-ELM, the multi-hidden layers structure of ML-DOS-ELM is able to help it extract more abundant features from the training data, and then helps the model to make more accurate predictions.

Algorithm 3 ML-DOS-ELM Algorithm

Input: A training dataset $D = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m\}_{i=1}^{N}$, the number of hidden layers q. Set the initial forgetting factor $\lambda = 1$ and the counter $k = 0$.

Output: The weights between the last hidden layer and output layer (i.e., output weights), marked as β.

Initialization phase:

Given a training dataset $D_0 = \{(x_i, t_i)|x_i \in R^d, t_i \in R^m\}_{i=1}^{N_0}$,

1) **For** $t = 1 \rightarrow q$ **do**

$$H_0^t = G((\beta_t^{(0)})^T H_0^{t-1})$$

where $G(\cdot)$ is the activation function and $\beta_t^{(0)}$ is the output weights for t_{th} layer, which is obtained by using (17).

end for

2) Calculate the initial output weights $\beta^{(0)} = P_0 H_0^T Y_0$, where $H_0 = H_0^q$ and $P_0 = (H_0^T H_0)^{-1}$.

3) Calculate the prediction accuracy of the initial model and mark it as ACC_0.

Online sequential learning phase:

4) When the $(k+1)_{th}$ chunk of new data D_{k+1} arrives,

For $t = 1 \rightarrow q$ **do**

$$H_{k+1}^t = G((\beta_t^{(k+1)})^T H_{k+1}^{t-1})$$

where $\beta_t^{(k+1)}$ is obtained by using (18).

end for

5) Update the output weights of ML-DOS-ELM by

$$\begin{aligned} P_{k+1} &= \lambda^{-2} P_k - \lambda^{-4} P_k H_{k+1}^T (I + \lambda^{-2} H_{k+1} P_k H_{k+1}^T)^{-1} \\ &\quad \times H_{k+1} P_k \\ \beta^{(k+1)} &= \beta^{(k)} + P_{k+1} H_{k+1}^T (T_{k+1} - H_{k+1} \beta^{(k)}) \end{aligned}$$

where $H_{k+1} = H_{k+1}^q$.

6) Calculate the prediction accuracy of the current model and mark it as ACC_{k+1}. Then calculate the accuracy change of the model by $E = ACC_{k+1} - ACC_k$ and update the forgetting factor λ by using (16).

7) Let $k = k + 1$ and go to 4) until all the new data are learned.

IV. PERFORMANCE EVALUATION

In this section, experimental results of the proposed algorithms on the classification, regression, time series, and image processing problems are presented. All the experiments are conducted in the MATLAB R2014b environment on the same Windows 7 OS with Intel Core i7-6700 3.40 GHz CPU and 32 GB RAM. In this paper, we use the *Sigmoid* function (i.e., $G(\omega, b, x) = 1/(\exp(-(\omega \cdot x + b)))$ as the activation function in all the cases. For each case, we compare the performance of each algorithm with different numbers of hidden layer nodes

TABLE 1. Details of the classification datasets.

Dataset	Attribute Number	Class Number	Sample Number	Drift Time	Hidden Nodes
Pima	8	2	768	Unclear	25
Credit	6	2	690	Unclear	25
Sonar	60	2	208	Unclear	25
Musk	166	2	476	Unclear	25
Spambase	57	2	458	Unclear	25
page	10	3	532	Unclear	10
Sea	3	3	60000	$\{15000\times i\}_{i=1}^{3}$	50
Rotating Hyperplane	10	2	90000	$\{10000\times i\}_{i=1}^{8}$	100
User1	100	2	1500	$\{300\times i\}_{i=1}^{5}$	100
User2	100	2	1500	$\{300\times i\}_{i=1}^{5}$	100

(the range is [1, 100]) and choose the best parameters for each algorithm. Notice that the tables show the averages of the results obtained by each algorithm with the best parameters over 50 trials.

A. CLASSIFICATION PROBLEMS

The performance of DOS-ELM is tested on 10 classification datasets. These datasets can be grouped into two categories: one is the dataset with unclear non-stationary properties, including Pima, Credit, Sonar, Musk, Spambase, and Page, which are selected from the UCI repository [29]; another is the dataset with clear non-stationary properties, including Sea [30], Hyperplane [31], Usenet1 [32], and Usenet2 [32]. The details of these datasets are shown in TABLE 1, including the number of attributes, classes, and samples, the moment when the distribution of the data stream changes (i.e., drift time), and the number of the hidden layer nodes in our experiments.

For each dataset, the training data and testing data are divided into 7:3, and the number of the initial samples in the training data is 50%. Since the number of samples in the first type of dataset (i.e., datasets with unclear non-stationary properties) is much smaller than that of the second type of dataset (i.e., datasets with clear non-stationary properties), the batch size of the first six datasets is set to [20, 30] in the sequential learning phase of each algorithm, while the batch size of the last four datasets is set to [50, 100].

The performance of OS-ELM [7], TOS-ELM [18], and DOS-ELM on the above 10 classification datasets is shown in TABLE 2. The best experimental results on each dataset are bolded.

As observed from TABLE 2, DOS-ELM achieves higher testing accuracy than OS-ELM and TOS-ELM in all cases. For example, for the dataset Hyperplane (a popular data stream problem with clear non-stationary properties), the prediction accuracy of DOS-ELM model is 19.35% higher than that of OS-ELM model and 11.80% higher than that of TOS-ELM model.

$$(0.7844 - 0.6572)/0.6572 * 100\% = 19.35\% \quad (19)$$

$$(0.7844 - 0.7016)/0.7016 * 100\% = 11.80\% \quad (20)$$

TABLE 2. Performance of OS-ELM, TOS-ELM, and DOS-ELM on the classification problems.

Datasets	Algorithm	Accuracy		Learning time (s)
		Training	Testing	
Pima	OS-ELM	0.7879	0.7696	0.0090
	TOS-ELM	0.7895	0.7500	0.0624
	DOS-ELM	0.7875	**0.7701**	0.0106
Credit	OS-ELM	0.7754	0.7522	0.0056
	TOS-ELM	0.7826	0.7495	0.0156
	DOS-ELM	0.7764	**0.7586**	0.0112
Sonar	OS-ELM	0.8049	0.7250	0.0050
	TOS-ELM	0.8143	0.6912	0.0624
	DOS-ELM	0.8074	**0.7306**	0.0112
Musk	OS-ELM	0.7552	0.7010	0.0062
	TOS-ELM	0.7264	0.7007	0.0780
	DOS-ELM	0.7579	**0.7127**	0.0243
Spambase	OS-ELM	0.8607	0.8270	0.0022
	TOS-ELM	0.8540	0.8322	0.0468
	DOS-ELM	0.8575	**0.8362**	0.0209
Page	OS-ELM	0.8170	0.8045	0.0012
	TOS-ELM	0.7898	0.7764	0.0351
	DOS-ELM	0.8266	**0.8171**	0.0037
Sea	OS-ELM	0.7138	0.7558	0.5304
	TOS-ELM	0.8631	0.8620	0.6084
	DOS-ELM	0.8742	**0.8802**	0.6864
Hyperplane	OS-ELM	0.8199	0.6572	0.4836
	TOS-ELM	0.8043	0.7016	0.6396
	DOS-ELM	0.7597	**0.7844**	0.6552
User1	OS-ELM	0.6674	0.7867	0.0624
	TOS-ELM	0.6644	0.7467	0.0780
	DOS-ELM	0.6704	**0.8000**	0.1404
User2	OS-ELM	0.7785	0.7733	0.1092
	TOS-ELM	0.7630	0.7533	0.1404
	DOS-ELM	0.7793	**0.7933**	0.1560

The experimental results show that the proposed DOS-ELM has better generalization performance than OS-ELM and TOS-ELM. In addition, DOS-ELM can deal with both stationary and non-stationary data stream problems through a unified framework. Taking the unstable dataset Sea as an example, the change of the forgetting factor in the update process of DOS-ELM is shown in FIGURE 2.

As shown in FIGURE 2, DOS-ELM can dynamically adjust the value of forgetting factor during the updating phase of the model, which can make it better adapt to the new data. For OS-ELM, the value of forgetting factor can be considered as a constant 1. Therefore, when the distribution of data streams changes, it is difficult for OS-ELM to make corresponding adjustments quickly, resulting in a lower accuracy. For TOS-ELM, although it can adjust the weight of the

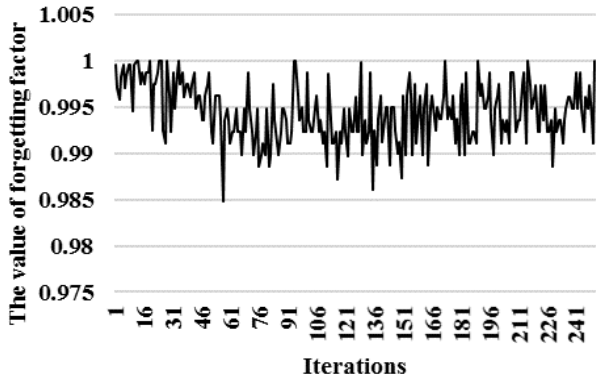


FIGURE 2. The change of the forgetting factor in the update process of the DOS-ELM model (dataset: Sea).

TABLE 3. Details of the regression datasets.

Dataset	Attributes	Sample Number	Hidden Nodes
Housing	13	506	50
Concrete Compressive Strength	8	1030	25
White Wine Quality	11	4898	25
Auto MPG	8	392	50
Red Wine Quality	11	1599	50

past data and new data according to the change of mean and variance of data stream, the above experimental results show that the effect of its adjustment strategy is not as good as that of DOS-ELM. One possible reason is that outliers or noise samples in data streams have a greater impact on the mean and variance, which makes its robustness poor.

B. REGRESSION PROBLEMS

For regression cases, five benchmark problems from UCI database [29] have been considered, that is, Housing, Concrete Compressive Strength, White Wine Quality, Auto MPG, and Red Wine Quality. The details of these datasets are shown in TABLE 3. The division ratio between the training and testing data in each dataset and the size of the chunk in the sequence learning phase of each algorithm are the same as the classification case.

For regression problems, we use the testing RMSE (Root Mean Square Error), testing SD (the Standard Deviation of testing errors), and the learning time as the performance indexes to evaluate the performance of OS-ELM [7], TOS-ELM [18], WOS-ELM [19], and DOS-ELM. The RMSE and SD are calculated by the following formulas.

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(o_i - t_i)^2}{N}} \tag{21}$$

$$SD = \sqrt{\frac{\sum_{j=1}^{K} (e_j - \bar{e})^2}{K - 1}} \tag{22}$$

TABLE 4. The Performance of OS-ELM, TOS-ELM, WOS-ELM, and DOS-ELM on the regression problems.

Datasets	Algorithm	SD	Average RMSE		Learning time (s)
			Training	Testing	
Housing	OS-ELM	0.0034	0.0141	0.0214	0.0109
	TOS-ELM	0.0059	0.0174	0.0251	0.0200
	WOS-ELM	0.0049	0.0140	0.0218	0.0764
	DOS-ELM	**0.0027**	0.0140	**0.0206**	0.0864
Concrete Compre Strength	OS-ELM	0.0009	0.0172	0.0185	0.0075
	TOS-ELM	0.0010	0.0173	0.0188	0.0159
	WOS-ELM	0.0010	0.0176	0.0187	0.0384
	DOS-ELM	**0.0008**	0.0171	**0.0183**	0.0577
White Wine Quality	OS-ELM	0.0353	0.0131	0.0307	0.0293
	TOS-ELM	0.0448	0.0229	0.0426	0.0362
	WOS-ELM	0.0270	0.0133	0.0276	0.1548
	DOS-ELM	**0.0184**	0.0131	**0.0259**	0.1576
Auto MPG	OS-ELM	0.2716	0.0460	0.0467	0.0031
	TOS-ELM	0.4434	0.1357	0.1361	0.0624
	WOS-ELM	1.4971	0.3106	0.3106	0.0577
	DOS-ELM	**0.0752**	0.0212	**0.0219**	0.0708
Red Wine Quality	OS-ELM	0.0059	0.0521	0.0661	0.0100
	TOS-ELM	0.0115	0.0585	0.0750	0.0331
	WOS-ELM	0.0096	0.0565	0.0725	0.2040
	DOS-ELM	0.0061	0.0522	**0.0659**	0.2103

where o_i is the prediction label of each testing sample, t_i is the real label of the corresponding sample, and N is the number of samples. In addition, K is the number of independent experiments for each case, e is the prediction error of the model in each experiment, and $\bar{e}$ is the average value of the prediction errors.

For WOS-ELM, the hyperparameters (i.e., the setpoint error ε_{fe} and the minimum of the forgetting factor $\lambda_{\min}$) are assigned using the same method as [19]. Based on the 5-fold cross-validation, the setpoint error and the minimum of the forgetting factor are set to $\varepsilon_{fe} = 0.0125$ and $\lambda_{\min} = 0.950$, respectively. The initial value of the forgetting factor is set to 1 in both the DOS-ELM and WOS-ELM.

The performance of OS-ELM, TOS-ELM, WOS-ELM, and DOS-ELM on the regression problems is given in TABLE 4.

As observed in TABLE 4, DOS-ELM has the smallest testing RMSE and SD in all the cases. For example, for the dataset AutoMPG, the testing RMSE of DOS-ELM model is 53.10% lower than that of OS-ELM model, 83.91% lower than that of TOS-ELM model, and 92.95% lower than that of WOS-ELM model.

$$(0.0467 - 0.0219)/0.0467 * 100\% = 53.10\% \tag{23}$$

$$(0.1361 - 0.0219)/0.1361 * 100\% = 83.91\% \tag{24}$$

$$(0.3106 - 0.0219)/0.3106 * 100\% = 92.95\% \tag{25}$$

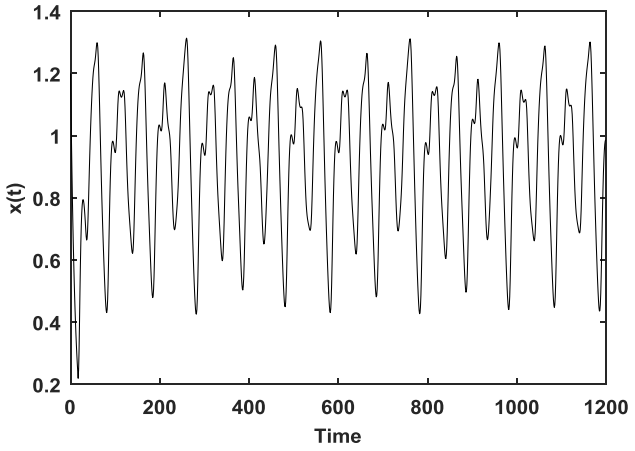


FIGURE 3. The Mackey-Glass chaotic time series problem.

In addition, the testing SD of DOS-ELM model is 72.31% lower than that of OS-ELM model, 83.04% lower than that of TOS-ELM model, and 94.98% lower than that of WOS-ELM model.

$$(0.2716 - 0.0752)/0.2716 * 100\% = 72.31\% \quad (26)$$
$$(0.4434 - 0.0752)/0.4434 * 100\% = 83.04\% \quad (27)$$
$$(1.4971 - 0.0752)/1.4971 * 100\% = 94.98\% \quad (28)$$

The experimental results show that the proposed DOS-ELM can achieve better generalization performance and stability than OS-ELM, TOS-ELM, and WOS-ELM.

C. TIME-SERIES PROBLEM

Time-series problems exist widely in the real world such as the stock market price prediction. In this section, we choose a classical time series problem, that is, the Mackey-Glass chaotic time series problem [33], to test the effectiveness of DOS-ELM. The details of the Mackey-Glass chaotic time series data can be visualized as FIGURE 3.

The Mackey-Glass chaotic time series data are generated by a time-delay differential equation

$$\frac{dx(t)}{dt} = \frac{0.2x(t-17)}{1+x(t-17)^{10}} - 0.1x(t) \quad (29)$$

where $x(t)$ denotes the output of the model at time t. We set $x(0) = 1.2$ and let $x(t) = 0$ if $t < 0$.

Similar to the data preprocessing methods in [7] and [19], the four past data that with the same time interval are used as the input to predict the current value. In our experiments, we collect 1000 instances from $t = 124$ to $t = 1123$, the input and output of the i_{th} sample can be represented as

$$X_i = [x(t-24),\ x(t-18),\ x(t-12),\ x(t-6)]^T$$
$$y_i = x(t)$$

The experimental results of OS-ELM, TOS-ELM, WOS-ELM, and DOS-ELM on the Mackey-Glass time series problem are given in TABLE 5.

TABLE 5. The Performance of OS-ELM, TOS-ELM, WOS-ELM, and DOS-ELM on the Mackey-Glass Time Series Problem.

Dataset	Algorithm	SD	Average RMSE		Learning time (s)
			Training	Testing	
Mackey Glass	OS-ELM	0.0020	0.0142	0.0139	0.0250
	TOS-ELM	0.0027	0.0140	0.0138	0.0499
	WOS-ELM	0.0026	0.0137	0.0134	0.0562
	DOS-ELM	**0.0013**	0.0131	**0.0128**	0.0624

As observed in TABLE 5, DOS-ELM achieves the smallest testing RMSE and testing standard deviation. For example, compared with OS-ELM, TOS-ELM, and WOS-ELM, the prediction error of the DOS-ELM model is 7.91% lower than that of the OS-ELM model, 7.25% lower than that of the TOS-ELM model, and 4.48% lower than that of the WOS-ELM model.

$$(0.0139 - 0.0128)/0.0139 * 100\% = 7.91\% \quad (30)$$
$$(0.0138 - 0.0128)/0.0138 * 100\% = 7.25\% \quad (31)$$
$$(0.0134 - 0.0128)/0.0134 * 100\% = 4.48\% \quad (32)$$

Similar phenomena also occur in the testing standard deviation (SD). The testing SD of DOS-ELM model is 35% lower than that of OS-ELM model, 51.85% lower than that of TOS-ELM model, and 50% lower than that of WOS-ELM model.

$$(0.0020 - 0.0013)/0.0020 * 100\% = 35.00\% \quad (33)$$
$$(0.0027 - 0.0013)/0.0027 * 100\% = 51.85\% \quad (34)$$
$$(0.0026 - 0.0013)/0.0026 * 100\% = 50.00\% \quad (35)$$

Therefore, we can infer that DOS-ELM has better generalization performance and stability than OS-ELM, TOS-ELM, and WOS-ELM on the time series problem.

D. IMAGE PROCESSING PROBLEM

In order to enhance the feature extraction ability of DOS-ELM, we extend it to a deep online model named ML-DOS-ELM, which has more than one hidden layer in its network structure. In this section, we use the famous face recognition dataset UMIST [34] and a popular handwritten digit recognition dataset MNIST [46] to verify the effectiveness of ML-DOS-ELM.

UMIST dataset contains 565 images taken from 20 different people and each of them is a 112×92 grayscale image. In our experiments, 400 images are used as the training data and the remaining 165 images are used as the testing data. For the MNIST dataset, there are 60000 samples for training, 10000 samples for testing, and the number of categories is 10. Each sample is a 28×28 grayscale image.

The performance of two deep online models (i.e., ML-DOS-ELM and ML-OS-ELM [20]) and two shallow online models (i.e., DOS-ELM and OS-ELM) are tested on the UMIST and MNIST. For consistency, both of the ML-DOS-ELM and ML-OS-ELM are set to be a

TABLE 6. The Performance of ML-DOS-ELM, ML-OS-ELM, DOS-ELM, and OS-ELM on the UMIST dataset.

Datasets	Algorithm	Accuracy		Learning time (s)
		Training	Testing	
UMIST	OS-ELM	0.9766	0.9486	178.2861
	DOS-ELM	0.9775	0.9543	181.1214
	ML-OS-ELM	0.9898	0.9789	195.7832
	ML-DOS-ELM	0.9991	**0.9886**	196.5623
MNIST	OS-ELM	0.9781	0.9679	957.1049
	DOS-ELM	0.9790	0.9686	963.8564
	ML-OS-ELM	0.9850	0.9713	1637.4957
	ML-OS-NNRW	0.9882	**0.9818**	1788.3250

three-hidden-layer architecture (3000-4000-5000). For DOS-ELM and OS-ELM, the number of the hidden layer nodes is to 5000. TABLE 6 shows the prediction accuracy of ML-DOS-ELM, ML-OS-ELM, DOS-ELM, and OS-ELM on the UMIST and MNIST.

As observed in TABLE 6, ML-DOS-ELM achieves the highest testing accuracy on the face recognition problem and the handwritten digit recognition problem. ML-DOS-ELM and ML-OS-ELM both have higher training and testing accuracy than OS-ELM and DOS-ELM, which implies that the deep models can extract more useful features from the training data. However, the training process of the deep models often takes longer than the shallow models due to the complexity of the network structure. In addition, it can be observed that the proposed shallow model DOS-ELM has higher prediction accuracy than OS-ELM, which implies that the proposed dynamic adjustment strategy of the forgetting factor also works in this case.

Regarding the question of why ML-DOS-ELM has a slightly higher prediction accuracy than ML-OS-ELM, here we give a speculative explanation. The common feature of ML-DOS-ELM and ML-OS-ELM is that they both use OS-ELM-AE to extract features. The output of the hidden layer of each previous OS-ELM-AE unit acts as the input of the next OS-ELM-AE, which enables the model to process and extract data features at multiple levels. From this process, it can be inferred that there is a cumulative problem of reconstruction errors in the process of feature extraction using OS-ELM-AE. Specifically, when generating the hidden layer features of the deep neural network by using the decoding parameters of OS-ELM-AE, the reconstruction error of each OS-ELM-AE unit still exists. As the number of hidden layers increases, the cumulative effect of the reconstruction error becomes more obvious. In this case, even if the concept drift does not occur in the training dataset, the error of the model may also increase with the increase of network complexity. By dynamically adjusting the forgetting factor, we can make the model adjust the relative importance of the past data and new data in time when the cumulative error and model error fluctuate greatly, which alleviates this problem to some extent. The dynamic adjustment strategy of the forgetting factor actually adds a regularizer to ML-OS-ELM. Therefore, even if the training dataset does not have clear non-stationary properties such as UMIST and MNIST, the experimental results show that ML-DOS-ELM has better prediction ability than ML-OS-ELM.

E. REMARKS

The above experimental results show that DOS-ELM can handle the data stream problems with clear non-stationary properties and stationary properties, which implies the great potential of DOS-ELM in dealing with data stream problems. However, several critical issues still remain.

Firstly, it is still an open problem to analyze the reasonability of DOS-ELM from the theoretical point of view.

Secondly, for different types of conceptual drift, such as the gradual drift and sudden drift, the dynamic adjustment strategy of forgetting factor may have different influences on the performance of the model. Noise samples in the data stream may have a negative impact on the performance of DOS-ELM. Another problem is that the new data may not have a strong relationship with the past data, which is similar to the issue of sudden drift. However, there is currently no corresponding countermeasures in DOS-ELM, which need to be further optimized.

Thirdly, for ML-DOS-ELM, the structure of multiple hidden layers enables it to extract features in a better way than the single-hidden layer DOS-ELM on some complex problems. The dynamic forgetting factor makes ML-DOS-ELM achieve better performance than other online deep models such as ML-OS-ELM. However, there is no free lunch in the world. The multi-hidden layers structure and the dynamic adjustment of the forgetting factor also cause ML-DOS-ELM to take longer to learn. The trade-off between the performance and complexity of the model is where we should continue to optimize.

Compared with the offline deep learning, online deep learning suffers more difficulties, such as the difficulty of extracting high-level features from a data stream with very few training samples. In other words, if the number of samples in the coming data chunk is very small, where an extreme case is that only one sample is coming at a time, the stability of the online deep learning algorithm is difficult to achieve. In many practical applications, one can accumulate the coming samples until the size of the data chunk reaches a certain threshold and then process them together. In this way, one can guarantee the stability of the deep online model to a certain extent. Since the threshold is unlikely to be large all the time, it may not have significant negative impact on the efficiency of the online model. For example, in our experiments, we found that ML-DOS-ELM could perform well once the size of the data chunk exceeded 20. However, the problem remains in some extreme cases, such as the applications which require that the model should only deal with samples one by one. We remark that it is worthy studying

how to ensure the stability of the online deep learning model in these cases.

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we propose a novel non-iterative online sequential learning algorithm named DOS-ELM and extend it to a deep online model named ML-DOS-ELM. In DOS-ELM, the forgetting factor is used for balancing the relative importance of new data and past data and is adjusted automatically and dynamically according to the accuracy change of the model in each learning epoch. Compared with the other online learning algorithms with forgetting factors or sliding windows, the advantages of DOS-ELM are as follows.

1) DOS-ELM can handle well data stream problems with both clear non-stationary properties and stationary properties.
2) DOS-ELM provides a unified framework for dealing with classification, regression, time series, and image processing problems.
3) Compared with other online learning algorithms with dynamic adjustment mechanism of the forgetting factor such as TOS-ELM, DOS-ELM is more robust to outliers and noise samples in the data stream and the computational complexity of the model is lower. Compared with WOS-ELM, the adjustment method of the forgetting factor in DOS-ELM avoids the influence of human factors on the performance of the model.
4) Besides, we extend DOS-ELM to ML-DOS-ELM, which has multiple hidden layers. ML-DOS-ELM inherits the advantages of DOS-ELM and can extract more high-level features from the training data due to the multi-hidden layers network structure.

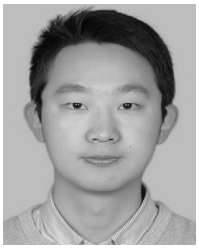
Extensive experiments demonstrate that DOS-ELM has better generalization performance than OS-ELM, TOS-ELM, and WOS-ELM. The experimental results on a face recognition problem and a handwritten digit recognition problem also show that ML-DOS-ELM can achieve higher prediction accuracy than DOS-ELM and the other deep online models such as ML-OS-ELM.

In the future, we will investigate the reasonability of DOS-ELM from a theoretical perspective and design a divide-and-conquer method based on DOS-ELM to deal with different types of concept drift such as gradual drift, recurrent drift, and sudden drift. Besides, we will also study new methods to deal with noise samples in the data stream and improve the robustness of DOS-ELM.

REFERENCES

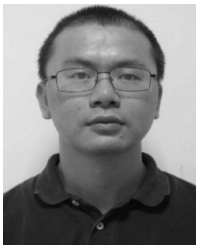
[1] A. Graves, A. R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.
[2] Z. Huang, Y. Yu, J. Gu, and H. Liu, "An efficient method for traffic sign recognition based on extreme learning machine," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 920–933, Apr. 2017.
[3] Y. Yang and Q. M. J. Wu, "Multilayer extreme learning machine with subnetwork nodes for representation learning," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2570–2583, Nov. 2016.
[4] Z. Y. Guan, L. Chen, W. Zhao, Y. Zheng, S. L. Tan, and D. Cai, "Weakly-supervised deep learning for customer review sentiment classification," in *Proc. IJCAI*, 2016, pp. 3719–3725.
[5] Y. Lecun, L. Bottou, G. B. Orr, and K. R. Müller, "Efficient backprop," *Neural Netw. Tricks Trade*, vol. 1524, no. 1, pp. 9–50, 1998.
[6] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 6, pp. 2284–2292, Dec. 2004.
[7] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.
[8] W. P. Cao, J. Z. Gao, Z. Ming, S. B. Cai, and Z. G. Shan, "Fuzziness based online sequential extreme learning machine for classification problems," *Soft Comput.*, vol. 22, no. 11, pp. 3487–3494, 2018.
[9] W. Cao, X. Wang, Z. Ming, and J. Gao, "A review on neural networks with random weights," *Neurocomputing*, vol. 275, pp. 278–287, Jan. 2018.
[10] W. T. Mao, J. W. Wang, L. He, and Y. Y. Tian, "Online sequential prediction of imbalance data with two-stage hybrid strategy by extreme learning machine," *Neurocomputing*, vol. 261, pp. 94–105, Oct. 2017.
[11] X. L. Jiang, J. F. Liu, Y. Q. Chen, D. J. Liu, Y. Gu, and Z. Y. Chen, "Feature adaptive online sequential extreme learning machine for lifelong indoor localization," *Neural Comput. Appl.*, vol. 27, no. 1, pp. 215–225, 2016.
[12] P. K. Wong, C. W. Hang, C. M. Vong, Z. C. Xie, and S. J. Huang, "Model predictive engine air-ratio control using online sequential extreme learning machine," *Neural Comput. Appl.*, vol. 27, no. 1, pp. 79–92, 2016.
[13] H. Wang and Z. Abraham, "Concept drift detection for streaming data," in *Proc. IJCNN*, 2015, pp. 1–9.
[14] S. Yu, Z. Abraham, H. Wang, M. Shah, Y. Wei, and J. C. Príncipe, "Concept drift detection and adaptation with hierarchical hypothesis testing," *J. Franklin Inst.*, vol. 356, no. 5, pp. 3187–3215, 2019.
[15] J. W. Zhao, Z. H. Wang, and D. S. Park, "Online sequential extreme learning machine with forgetting mechanism," *Neurocomputing*, vol. 87, pp. 79–89, Jun. 2012.
[16] X. Y. Wang and M. Han, "Online sequential extreme learning machine with kernels for nonstationary time series prediction," *Neurocomputing*, vol. 145, pp. 90–97, Dec. 2014.
[17] X. R. Zhou, Z. J. Liu, and C. X. Zhu, "Online regularized and kernelized extreme learning machines with forgetting mechanism," *Math. Problems Eng.*, vol. 3, pp. 231–237, Jul. 2014.
[18] Y. Gu, J. Liu, Y. Chen, X. Jiang, and H. Yu, "TOSELM: Timeliness online sequential extreme learning machine," *Neurocomputing*, vol. 128, pp. 27–119, Mar. 2014.
[19] H. G. Zhang, S. Zhang, and Y. Yin, "Online sequential ELM algorithm with forgetting factor for real applications," *Neurocomputing*, vol. 261, pp. 144–152, Oct. 2017.
[20] B. Mirza, K. Stanley, and F. Dong, "Multi-layer online sequential extreme learning machine for image classification," in *Proc. ELM*, vol. 1, 2016, pp. 39–49.
[21] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. IJCNN*, vol. 2, 2004, pp. 985–990.
[22] Y.-H. Pao and Y. Takefuji, "Functional-link net computing: Theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, May 1992.
[23] W. F. Schmidt, M. A. Kraaijveld, and R. P. Duin, "Feedforward neural networks with random weights," in *Proc. IAPR*, 1992, pp. 1–4.
[24] X. Liu, S. Lin, J. Fang, and Z. Xu, "Is extreme learning machine feasible? A theoretical assessment (Part I)," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 7–20, Jan. 2015.
[25] S. Lin, X. Liu, J. Fang, and Z. Xu, "Is extreme learning machine feasible? A theoretical assessment (Part II)," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 21–34, Jan. 2015.
[26] U. Muhammad and A. Mian, "Blind domain adaptation with augmented extreme learning machine features," *IEEE Trans. Cybern.*, vol. 47, no. 3, pp. 651–660, Mar. 2017.
[27] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2015.
[28] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1996.

[29] C. Blake and C. Merz, "UCI repository of machine learning databases," Dept. Inf. Comp. Sci., Univ. California, Irvine, CA, Tech. Rep. 01, May 1998. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html
[30] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proc. 7th Int. Conf. Knowl. Kiscovery Data Mining*, 2001, pp. 377–382.
[31] W. Fan, "Systematic data selection to mine concept-drifting data streams," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Kiscovery Data Mining*, 2004, pp. 128–137.
[32] I. Katakis, G. Tsoumakas, and I. P. Vlahavas, "An ensemble of classifiers for coping with recurring contexts in data Streams," in *Proc. Eur. Conf. Artif. Intell. (ECAI)*, 2008, pp. 763–764.
[33] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.
[34] D. B. Graham and N. M. Allinson, "Characterising virtual eigensignatures for general purpose face recognition," in *Face Recognition: From Theory to Applications*, vol. 163. Berlin, Germany: Springer, 1998, pp. 446–456.
[35] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with extreme learning machine for big data," *IEEE Intell. Syst.*, vol. 28, no. 6, pp. 31–34, Nov. 2013.
[36] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.
[37] Q. R. Li, Y. D. Wang, and X. F. Zhang, "Analysis and simulation of a variable forgetting factor RLS algorithm," *Mod. Electron. Techn.*, vol. 17, no. 1, pp. 45–47, 2008.
[38] D. Sahoo, Q. Pham, J. Lu, and S. Hoi, "Online deep learning: Learning deep neural networks on the fly," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 2660–2666.
[39] H. L. Yu and G. I. Webb, "Adaptive online extreme learning machine by regulating forgetting factor by concept drift map," *Neurocomputing*, vol. 343, pp. 141–153, May 2019.
[40] S. G. Soares and R. Araújo, "An adaptive ensemble of on-line extreme learning machines with variable forgetting factor for dynamic system prediction," *Neurocomputing*, vol. 171, pp. 693–707, Jan. 2016.
[41] D. Xiao, B. Li, and S. Zhang, "An online sequential multiple hidden layers extreme learning machine method with forgetting mechanism," *Chemometrics Intell. Lab. Syst.*, vol. 176, pp. 126–133, May 2018.
[42] J.-M. Park and J.-H. Kim, "Online recurrent extreme learning machine and its application to time-series prediction," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2017, pp. 1983–1990.
[43] W. Mao, L. He, Y. Yan, and J. Wang, "Online sequential prediction of bearings imbalanced fault diagnosis by extreme learning machine," *Mech. Syst. Signal Process.*, vol. 83, pp. 450–473, Jan. 2017.
[44] B. Mirza and Z. Lin, "Meta-cognitive online sequential extreme learning machine for imbalanced and concept-drifting data classification," *Neural Netw.*, vol. 80, pp. 79–94, Aug. 2016.
[45] B. Mirza, Z. Lin, and N. Liu, "Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift," *Neurocomputing*, vol. 149, pp. 316–329, Feb. 2015.
[46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[47] Y. Li, S. Zhang, Y. Yin, W. Xiao, and J. Zhang, "A novel online sequential extreme learning machine for gas utilization ratio prediction in blast furnaces," *Sensors*, vol. 17, no. 8, p. 1847, 2017.
[48] S. Ding, B. Mirza, Z. Lin, J. Cao, X. Lai, T. V. Nguyen, and J. Sepulveda, "Kernel based online learning for imbalance multiclass classification," *Neurocomputing*, vol. 277, pp. 139–148, Feb. 2018.
[49] T. V. Nguyen and B. Mirza, "Dual-layer kernel extreme learning machine for action recognition," *Neurocomputing*, vol. 260, pp. 123–130, Oct. 2017.
[50] J. Zhang, W. Xiao, Y. Li, and S. Zhang, "Residual compensation extreme learning machine for regression," *Neurocomputing*, vol. 311, pp. 126–136, Oct. 2018.
[51] J. Zhang, W. Xiao, Y. Li, S. Zhang, and Z. Zhang, "Multilayer probability extreme learning machine for device-free localization," *Neurocomputing*, to be published.



WEIPENG CAO received the Ph.D. degree in computer science from Shenzhen University, in June 2019. Since 2017, he has been serving as a Visiting Scholar with the School of Engineering and Computer Science, University of the Pacific, Stockton, CA, USA. He is currently an Associate Researcher with the College of Computer Science and Software Engineering, Shenzhen University. His research interests include machine learning and deep learning.

ZHONG MING received the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, China. He is currently a Professor with the College of Computer Science and Software Engineering, Shenzhen University, Guangdong, China. He has published more than 100 high-quality articles in top conferences and journals, such as the IEEE TRANSACTIONS ON COMPUTERS. His current research interests include cloud computing, the Internet of Things, software engineering, and artificial intelligence.



ZHIWU XU received the Ph.D. degree in computer science from University Paris Diderot–Paris 7 and the University of Chinese Academy of Sciences, under the joint cultivation, in 2013. He is currently an Assistant Professor with Shenzhen University. His research interests include in the area of program analysis and verification, type systems, software security, and machine learning.

JIYONG ZHANG received the B.S. and M.S. degrees in computer science from Tsinghua University, in 1999 and 2001, respectively, and the Ph.D. degree in computer science from the Swiss Federal Institute of Technology at Lausanne (EPFL), in 2008. He is currently a Distinguished Professor with Hangzhou Dianzi University. His research interests include intelligent information processing, machine learning techniques, data sciences, and recommender systems.

QIANG WANG received the Ph.D. degree from EPFL, Switzerland, in 2017. His research focuses on rigorous model-based system design, with the goal of providing efficient verification techniques and tools for embedded concurrent systems, in particular, the systems constructed using machine learning algorithms.