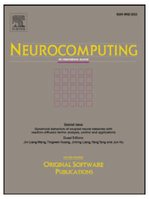
Neurocomputing 275 (2018) 2864–2879

Contents lists available at ScienceDirect

Neurocomputing

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



The memory degradation based online sequential extreme learning machine

Quan-Yi Zou[a], Xiao-Jun Wang[a,*], Chang-Jun Zhou[a,*], Qiang Zhang[a,b,*]

[a] Key Laboratory of Advanced Design and Intelligent Computing, Ministry of Education, Dalian University, Dalian 116622, China
[b] School of Computer Science & Technology, Dalian University of Technology, Dalian 116622, China

ARTICLE INFO

Article history:
Received 17 May 2017
Revised 22 September 2017
Accepted 11 November 2017
Available online 24 November 2017

Communicated by Yudong Zhang

Keywords:
Online learning
Extreme learning machine
Memory factor
Similarity

ABSTRACT

In online learning, the contribution of old samples to a model decreases as time passes, and old samples gradually become invalid. Although the Online Sequential Extreme Learning Machine (OS-ELM) can avoid the repetitive training of old samples, invalid samples are still used, which goes against improving the accuracy of an OS-ELM model. The Online Sequence Extreme Learning Machine with Forgetting Mechanism (FOS-ELM) timely discards invalid samples, but it does not consider the differences among valid samples and then has the limitation on boosting the accuracy and generalization. To solve this issue, the Memory Degradation Based OS-ELM (MDOS-ELM) is proposed in this paper. The MDOS-ELM adjusts the weights of the old and new samples in real time by a self-adaptive memory factor, and simultaneously discards invalid samples. The self-adaptive memory factor is determined by two elements. One is the similarity between the new and old samples, and the other is the prediction errors of the current training samples on the previous model. The performance of the proposed MDOS-ELM is validated on both regression and classification datasets which include an artificial dataset and twenty-two real-world dataset. The results demonstrate that the MDOS-ELM model outperforms the OS-ELM and the FOS-ELM models on the accuracy and generalization.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

The Single-hidden Layer Feed-forward Neural-network (SLFN) with a nonlinear activation function is a strong learning algorithm, which can approximate any nonlinear continuous function [1]. Since it was proposed, the SLFN has been widely concerned in both theory and practice [2–5]. Unfortunately, it is required to iteratively adjust the parameters for optimizing the SLFN model, which makes the SLFN lack of the fast learning ability.

To address this problem, based on the Moore-Penrose generalized inverse theory, the Extreme Learning Machine (ELM) is proposed by Huang et al. [6–8]. Compared with the traditional SLFN algorithm, the ELM only needs to set reasonable numbers of the hidden layer nodes and randomly assign parameters of the hidden layer and then the output weights of the ELM are obtained by the least squares method. Only one step is carried out in the whole learning process of the ELM. Namely, the ELM does not iteratively update all the parameters of the SLFN, which makes it have the advantage of fast learning.

However, the ELM has the following disadvantages. First, the ELM only considers the empirical risk minimization and easily causes the over-fitting issue. To solve this issue, the regularized ELM (RELM) [9] was proposed by Deng et al., which was based on both the structural risk and the empirical risk minimizations, but the RELM has the shortcoming which cannot cope with non-Gaussian noises [10]. Second, the ELM cannot handle the curse of dimensionality well. To handle this disadvantage, kernel functions were introduced into the ELM, which could map samples into higher dimensional feature space [11–13], but these algorithms have the shortcoming which is hard to select an optimal kernel function for a specific application. Third, it is hard to set a suitable number of nodes in hidden layer. If the number is too large, the computation of the ELM model is time-consuming, and inversely, and the ELM model is not accurate enough. To address this problem, literature [14–18] proposed strategy that gradually added or reduced the number of nodes in the hidden layer to optimize the structure of an ELM model, but the operation pruned or increased is time consuming. Fourthly, the ELM has the limitation in drawing deeper relationship among the data from too complicated

* Corresponding authors.
E-mail addresses: wxjjessicaxj0903@126.com (X.-J. Wang), zhouchangjun@dlu.edu.cn (C.-J. Zhou), zhangq26@126.com (Q. Zhang).

https://doi.org/10.1016/j.neucom.2017.11.030
0925-2312/© 2017 Elsevier B.V. All rights reserved.

practical problems. To deal with this disadvantage, the ELM was usually modified by combining with the deep learning algorithms [19], and then the multilayer ELM [20] and the deep ELM [21] were proposed. Additionally, many revisions of the ELM were presented and applied in various fields, such as the wind forecast [22,23], the image processing [24,25], the face recognition [26], the solar radiation prediction [27], the medical diagnosis [28,29], pathological brain detection [30–32], and the time series predictions [33].

The traditional ELM and most of its revisions are more suitable for offline learning. Prior to offline learning, a certain number of samples are accumulated, which are used to train a model and then the model is applied to obtain the prediction of a new sample. However, in many real applications, such as the price forecasting [34], the weather forecasting [35], and the robotic online controllers [36], the samples are updated in real time. If the ELM is directly used for the online learning, when the chunk of new samples is acquired, the ELM takes the old and new samples together as a new sample chunk. Then the new sample chunk is relearned by the ELM. As the accumulation of samples and the repeated learning of the old samples, the learning speed of the ELM will gradually decline, and even worse, the ELM may be unable to make the online learning continuous.

In order to effectively avoid the repeated learning of old samples, the Online Sequential ELM (OS-ELM) is proposed by Liang et al. [37], which gradually adds the new samples for learning chunk by chunk. By using the output weights of the previous OS-ELM model and the new samples, the OS-ELM updates the output weights and improves the speed of online learning. On the basis of the OS-ELM, the fuzzy OS-ELM based on fuzzy inference system is proposed by Rong et al. [38], and the OS-ELM with kernels is proposed by Scardapane et al. [39]. The OS-ELM and its improved algorithms randomly assign initial parameters of the hidden layer, which brings the instability of the models. To deal with the issue, the ensemble OS-ELMs is proposed by Lan et al. [40], which is a combination of several independent OS-ELM sub-models. Although the EOS-ELMs algorithm has strong stability, it is not prominent to improve the accuracy and the generalization. The reason is that it ignores the timeliness of samples.

The so-called timeliness of samples is that as time elapses old samples will be gradually invalid in online learning. Applying the invalid samples in online learning usually leads to the declining accuracy and generalization of the EOS-ELMs model. Aiming at this issue, the OS-ELM with Forgetting Mechanism (FOS-ELM) is proposed by Zhao et al. [41], in which the invalid samples are timely discarded by the forgetting mechanism. However, the differences among valid samples are ignored by the FOS-ELM. In an online learning process, the importance of the valid samples to a model is related with their age [42]. New samples are usually more important than old ones, which should be given greater weights, vice versa.

In order to fully utilize the differences among valid samples and to further improve the accuracy and generalization of the FOS-ELM model, the Memory Degradation based OS-ELM (MDOS-ELM) is proposed in this paper. On the one hand, the self-adaptive memory factor is introduced in the MDOS-ELM, which can dynamically adjust the weights of the old and new valid samples in the online learning processes. The self-adaptive memory factor is determined by two elements. One is the similarity between the input variables of the new and old samples, and the other is the predicting errors of the current training samples on the previous model. The former is from the point of view of the input variables, and the latter is from the point of view of output variable. There are many methods for evaluating the similarity of samples, such as Euclidean distance, Cosine, Pearson-Correlation coefficient [43] and Robust-Huber Similarity Measure (RHSM) [44]. The first three are common methods and they are too sensitive to outliers. With Huber function [45], the last one is more robustness to outliers.

On the other hand, the mechanism of timely discarding invalid samples in the FOS-ELM is referred to by the MDOS-ELM. The performance of the MDOS-ELM is verified on nine regression datasets and fourteen classification datasets. The accuracy and generalization of the MDOS-ELM is compared with those of the OS-ELM and the FOS-ELM.

The remainder of this paper is organized as follows. Section 2 introduces the ELM, the OS-ELM and the FOS-ELM algorithms. In Section 3, the MDOS-ELM algorithm is proposed and the self-adaptive memory factor is discussed. In Section 4, nine regression datasets and fourteen classification datasets are adopted in the experiments to verify the performance of the proposed MDOS-ELM. The performance of the MDOS-ELM is compared with that of the OS-ELM and the FOS-ELM. In Section 5, the conclusion of this paper is summarized.

2. Background

2.1. The ELM algorithm

The learning set is $S = \{(\mathbf{x}_i, t_i) | \mathbf{x}_i \in \mathrm{R}^n, t_i \in \mathrm{R}, i = 1, 2, ..., N\}$, where t is the output variable and $\mathbf{x} = [x_1, x_2, ..., x_n]$ is the input vector. N is the number of samples and n is the dimension of the input variables. A unified SLFN model with L nodes in the hidden layer can be expressed as follows [5],

$$f_L(\mathbf{x}) = \sum_{l=1}^{L} \beta_l G(\mathbf{a}_l, b_l, \mathbf{x}), \mathbf{x} \in \mathrm{R}^n, \tag{1}$$

where β_l is the output weight of the lth node of the hidden layer. $G(\cdot)$ is the activation function of the hidden layer node, and it can be ‘RBF’, ‘Sigmoid’, ‘Sine’, or ‘hradlim’ etc. $\mathbf{a}$ and b are parameters of $G(\cdot)$.

The output of the SLFN with L hidden layer nodes is expected to approximate the real output with zero errors, and then

$$t_i = \sum_{l=1}^{L} \beta_l G(\mathbf{a}_l, b_l, \mathbf{x}_i), i = 1, 2, \ldots, N.$$

The above N equations can be written in the matrix form below

$$\mathbf{T} = \mathbf{H}\beta,$$

where $\mathbf{T} = [t_1, t_2, ..., t_N]^T$ is the expected output matrix, and $\beta = [\beta_1, \beta_2, ..., \beta_L]^\mathrm{T}$ is the output weight. $\mathbf{H}$ is the output matrix of the hidden layer, and

$$\mathbf{H} = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{L \times N}.$$

$\hat{\beta}$ denotes the estimated value of $\boldsymbol{\beta}$, which is obtained by the least-square method. The relationship between $\hat{\beta}$ and $\mathbf{T}$ is as follows,

$$\hat{\beta} = \mathbf{H}^\dagger \mathbf{T},$$

where $\mathbf{H}$† is the Moore–Penrose generalized inverse of $\mathbf{H}$, and

$$\mathbf{H}^\dagger = \left(\mathbf{H}^\mathrm{T}\mathbf{H}\right)^{-1}\mathbf{H}^\mathrm{T}.$$

2.2. The OS-ELM algorithm

The OS-ELM is an online version of the ELM. It can gradually obtain new sample chunk (with fixed or chunk sizes). The OS-ELM algorithm is shown in Algorithm 1.

Algorithm 1
The OS-ELM algorithm [37].

The OS-ELM algorithm consists of two phases.

Phase I Initialization

The initial sample chunk $S_0 = \{(\mathbf{x}_i^{(0)}, t_i^{(0)}) | \mathbf{x}_i^{(0)} \in R^n, t_i^{(0)} \in R, i = 1, 2, ..., N_0\}$ contains N_0 samples, the number of hidden layer nodes is $L(L \le N_0)$, and $G(\cdot)$ is the activation function.

Step 1, parameters of the hidden layer **a** and b are randomly assigned. Generally, both **a** and b belong to $(-1, 1)$. When the 'RBF' is used as the activation function, b belongs to $(0, 1)$.

Step 2, the hidden layer output matrix is computed,

$$\mathbf{H}_0 = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1^{(0)}) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1^{(0)}) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_{N_0}^{(0)}) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_{N_0}^{(0)}) \end{bmatrix}_{L\times N_0}.$$

Step 3, the initial output weight $\boldsymbol{\beta}^{(0)}$ is calculated, and $\beta^{(0)} = \mathbf{P}_0\mathbf{H}_0^{\mathrm{T}}\mathbf{T}_0$, where $\beta^{(0)} = [\beta_1^{(0)}, \beta_2^{(0)}, ..., \beta_L^{(0)}]^T$, $\mathbf{P}_0 = (\mathbf{H}_0^{\mathrm{T}}\mathbf{H}_0)^{-1}$,and $\mathbf{T}_0 = [t_1^{(0)}, t_2^{(0)}, ..., t_{N_0}^{(0)}]^T$.

Phase II Online sequential learning

The $(k+1)-$th chunk of new training samples $S_{k+1} = \{(\mathbf{x}_i^{(k+1)}, t_i^{(k+1)}) | \mathbf{x}_i^{(k+1)} \in R^n,\ t_i^{(k+1)} \in R,\ i = 1, 2, \ldots, N_{k+1}\}$ contained N_{k+1} samples is added into learning.

Step 4, the output matrix $\mathbf{H}_{k+1}$ of the hidden layer is computed

$$\mathbf{H}_{k+1} = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1^{(k+1)}) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1^{(k+1)}) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_{N_0}^{(k+1)}) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_{N_0}^{(k+1)}) \end{bmatrix}_{L\times N_0}.$$

Step 5, the output weight $\beta^{(k+1)}$ of the hidden layer is computed

$$\beta^{(k+'1)} = \beta^{(k)} + \mathbf{P}_{k+1}\mathbf{H}_{k+1}^{\mathrm{T}}(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\beta^{(k)}),$$

where $\mathbf{T}_{k+1} = \left[t_1^{(k+1)}, t_2^{(k+1)}, \cdots, t_{N_{k+1}}^{(k+1)}\right]^T$ and $\mathbf{P}_{k+1} = \left(\sum_{j=1}^{k+1} \mathbf{H}_j^{\mathrm{T}}\mathbf{H}_j\right)^{-1}$. According to Wood-bury,

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k\mathbf{H}_{k+1}^{\mathrm{T}}(\mathbf{I} + \mathbf{H}_{k+1}\mathbf{H}_{k+1}^{\mathrm{T}})^{-1}\mathbf{H}_{k+1}\mathbf{P}_k.$$

Then, the current SLFN model can be obtained based on Eq. (1).

Step 6, when a new sample chunk is added, k is replaced by $k+1$ and then turn to Step 4.

Algorithm 2
The FOS-ELM algorithm [41].

The FOS-ELM algorithm consists of two phases.

Phase I Initialization

As mentioned above, S_0 is the initial sample chunk, L is the number of the hidden layer nodes, and $G(\cdot)$ is the activation function.

Step 1, the parameters of the hidden layer **a** and b are randomly assigned.

Step 2, the output matrix of the hidden layer $\mathbf{H}_0$ is computed.

Step 3, the initial output weight $\beta^{(0)} = \mathbf{P}_0\mathbf{H}_0^{\mathrm{T}}\mathbf{T}_0$ is computed, where $\beta^{(0)} = [\beta_1^{(0)}, \beta_2^{(0)}, ..., \beta_L^{(0)}]^{\mathrm{T}}$, $\mathbf{P}_0 = (\mathbf{H}_0^{\mathrm{T}}\mathbf{H}_0)^{-1}$ and $\mathbf{T}_0 = [t_1^{(0)}, t_2^{(0)}, ..., t_{N_0}^{(0)}]^{\mathrm{T}}$.

Phase II Online sequential learning phase

The $(k+1)-$th sample chunk S_{k+1} is obtained into learning.

Step 4, the output matrix $\mathbf{H}_{k+1}$ of the hidden layer is computed.

Step 5, the output weight is $\beta^{(k+1)}$ computed.

When $k+1 < u$,

$$\beta^{(k+1)} = \beta^{(k)} + \mathbf{P}_{k+1}\mathbf{H}_{k+1}^{\mathrm{T}}(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\beta^{(k)}),$$

where

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k\mathbf{H}_{k+1}^{\mathrm{T}}(\mathbf{I} + \mathbf{H}_{k+1}\mathbf{H}_{k+1}^{\mathrm{T}})\mathbf{H}_{k+1}\mathbf{P}_k.$$

When $k+1 \ge u$, the $(k-u+1)-$th sample chunk will become invalid.

$$\beta^{(k+1)} = \beta^{(k)} + \mathbf{P}_{k+1}\begin{bmatrix} -\mathbf{H}_{k-u+1} \\ \mathbf{H}_{k+1} \end{bmatrix}(\begin{bmatrix} \mathbf{T}_{k-u+1} \\ \mathbf{T}_{k+1} \end{bmatrix} - \begin{bmatrix} \mathbf{H}_{k-u+1} \\ \mathbf{H}_{k+1} \end{bmatrix}\beta^{(k+1)}),$$

where

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k\begin{bmatrix} -\mathbf{H}_{k-u+1} \\ \mathbf{H}_{k+1} \end{bmatrix}^{\mathrm{T}}(\mathbf{I} + \begin{bmatrix} \mathbf{H}_{k-u+1} \\ \mathbf{H}_{k+1} \end{bmatrix}\mathbf{P}\begin{bmatrix} -\mathbf{H}_{k-u+1} \\ \mathbf{H}_{k+1} \end{bmatrix}^{\mathrm{T}})^{-1}\begin{bmatrix} \mathbf{H}_{k-u+1} \\ \mathbf{H}_{k+1} \end{bmatrix}\mathbf{P}_k.$$

Then, the current SLFN model can be obtained based on Eq. (1).

Step 6, when a new sample chunk is obtained, k is replaced by $k+1$ and then turn to Step 4.

2.3. The FOS-ELM algorithm

The FOS-ELM assumes that a sample chunk is added into learning in each unit time. The validity period of each chunk is u unit time. Exceeding the validity period, samples will be discarded. The FOS-ELM algorithm is shown in Algorithm 2.

3. The proposed MDOS-ELM algorithm

3.1. The MDOS-ELM

During the whole online learning process, the old and new samples are equally treated by the OS-ELM and the FOS-ELM. However, differences are existed between the old and new samples. The new samples can better reflect the characteristics and trends of the current data than the old ones. Additionally, each sample chunk has a period of validity. Sample chunk will become invalid if its validity time exceeds period of validity.

Considering the differences and timeliness of the samples, the MDOS-ELM is proposed in this work. On the basis of the FOS-ELM, a self-adaptive memory factor is introduced into the MDOS-ELM. The self-adaptive memory factor is determined by two elements. One is the similarity between the new and old samples, and the other is the prediction errors of the current training samples on the previous model. The whole learning process can be understood in this way, the memory ability of the MDOS-ELM to samples continuously decreases over time. And the closer to the present time the deeper the memory is, the larger the weight, vice versa.

When the period of validity is exceeded, the memory ability of the MDOS-ELM to samples is reduced to zero, and the samples will be discarded.

After the initial sample chunk $S_0 = \{(\mathbf{x}_i^{(0)}, t_i^{(0)})|\mathbf{x}_i^{(0)} \in \mathrm{R}^n, t_i^{(0)} \in \mathrm{R}, i = 1, 2, ..., N_0\}$is learned, the initial output weights $\boldsymbol{\beta}^{(0)}$ are obtained, and $\beta^{(0)} = \mathbf{P}_0\mathbf{H}_0^{\mathrm{T}}\mathbf{T}_0$, where $\mathbf{P}_0 = (\mathbf{H}_0^{\mathrm{T}}\mathbf{H}_0)^{-1}$. It is assumed that each sample chunk is added into learning in each unit time. The validity period of any sample chunk is u unit time. The valid range of the k-th sample chunk $S_k = \{(\mathbf{x}_i^{(k)}, t_i^{(k)})|\mathbf{x}_i^{(k)} \in \mathrm{R}^n, t_i^{(k)} \in \mathrm{R}, i = 1, 2, ..., N_k\}, k = 0, 1, 2, \ldots \infty$ is from the k-th unit time to the $(k+u)$-th unit time.

When a new sample chunk S_1 is acquired, the output matrix of hidden layer $\mathbf{H}_1$ is computed. S_1 can better reflect the characteristics and trends of the current data than S_0. Therefore, the weight of S_1 in a MDOS-ELM model should be larger than that of S_0. In order to reflect the difference between the old and new samples, the memory factor λ_0, $0 < \lambda_0 < 1$ is introduced to adjust the weights of S_1 and S_0

$\boldsymbol{\beta}^{(1)}$ represents the output weights, and

$$\beta^{(1)} = \left(\lambda_0\mathbf{H}_0^{\mathrm{T}}\mathbf{H}_0 + \mathbf{H}_1^{\mathrm{T}}\mathbf{H}_1\right)^{-1}\left(\lambda_0\mathbf{H}_0^{\mathrm{T}}\mathbf{T}_0 + \mathbf{H}_1^{\mathrm{T}}\mathbf{T}\right), \tag{2}$$

where $\mathbf{T}_1 = \left[t_1^{(1)}, t_2^{(1)}, \ldots, t_{N_1}^{(1)}\right]^{\mathrm{T}}$.

Let $\mathbf{L}_1 = \lambda_0\mathbf{H}_0^{\mathrm{T}}\mathbf{H}_0 + \mathbf{H}_1^{\mathrm{T}}\mathbf{H}_1$ and $\mathbf{P}_1 = \mathbf{L}_1^{-1}$. Now Eq. (2) can be transformed into

$$\begin{aligned}\beta^{(1)} &= \mathbf{L}_1^{-1}\left(\lambda_0\mathbf{H}_0^{\mathrm{T}}\mathbf{T}_0 + \mathbf{H}_1^{\mathrm{T}}\mathbf{T}_1\right)\\ &= \mathbf{P}_1\left(\lambda_0\mathbf{H}_0^{\mathrm{T}}\mathbf{T}_0 + \mathbf{H}_1^{\mathrm{T}}\mathbf{T}_1\right)\\ &= \beta^{(0)} + \mathbf{P}_1\mathbf{H}_1^{\mathrm{T}}\left(\mathbf{T}_1 - \mathbf{H}_1\beta^{(0)}\right).\end{aligned}$$

When the k-th chunk of samples S_k is acquired, the output matrix of hidden layer $\mathbf{H}_k$ is calculated, and then the output weight is updated. $\overline{S_k}$ denotes the current valid sample set. When $k < u$, all the validity periods are not exceeded, and $\overline{S_k} = \bigcup_{i=0}^{k} S_i$.. And $\mathbf{L}_k$ is analogized as follows,

$$\mathbf{L}_k = \lambda_{k-1}\mathbf{L}_{k-1} + \mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k. \tag{3}$$

Then, $\boldsymbol{\beta}^{(k)}$ is denoted as follows,

$$\begin{aligned}\beta^{(k)} &= \mathbf{L}_k^{-1}\left(\lambda_{k-1}\mathbf{H}_{k-1}^{\mathrm{T}}\mathbf{T}_{k-1} + \mathbf{H}_k^{\mathrm{T}}\mathbf{T}_k\right)\\ &= \mathbf{P}_k\left(\lambda_{k-1}\mathbf{H}_{k-1}^{\mathrm{T}}\mathbf{T}_{k-1} + \mathbf{H}_k^{\mathrm{T}}\mathbf{T}_k\right)\\ &= \beta^{(k-1)} + \mathbf{P}_k\mathbf{H}_k^{\mathrm{T}}\left(\mathbf{T}_k - \mathbf{H}_k\beta^{(k-1)}\right),\end{aligned} \tag{4}$$

where $\mathbf{T}_k = [t_1^{(k)}, t_2^{(k)}, ..., t_{N_1}^{(k)}]^{\mathrm{T}}$ and $\mathbf{P}_k = \mathbf{L}_k^{-1}$. With the Wood-bury formula [46] employed, the update formula of $\mathbf{P}_k$ is derived, and

$$\begin{aligned}\mathbf{P}_k &= \left(\lambda_{k-1}\mathbf{L}_{k-1} + \mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k\right)^{-1}\\ &= (\lambda_{k-1}\mathbf{L}_{k-1})^{-1} - (\lambda_{k-1}\mathbf{L}_{k-1})^{-1}\mathbf{H}_k^{\mathrm{T}}\left(\mathbf{I} + \mathbf{H}_k\left(\lambda_{k-1}\mathbf{L}_{k-1}\right)^{-1}\mathbf{H}_k^{\mathrm{T}}\right)^{-1}\\ &\quad\times \mathbf{H}_k(\lambda_{k-1}\mathbf{L}_{k-1})^{-1}\\ &= \frac{1}{\lambda_{k-1}}(\mathbf{L}_{k-1})^{-1} - \frac{1}{\lambda_{k-1}^2}\mathbf{L}_{k-1}^{-1}\mathbf{H}_k^{\mathrm{T}}\left(\mathbf{I} + \frac{1}{\lambda_{k-1}}\mathbf{H}_k\mathbf{L}_{k-1}^{-1}\mathbf{H}_k^{\mathrm{T}}\right)^{-1}\mathbf{H}_k\mathbf{L}_{k-1}^{-1}\\ &= \frac{1}{\lambda_{k-1}}\left(\mathbf{L}_{k-1}^{-1} - \mathbf{L}_{k-1}^{-1}\mathbf{H}_k^{\mathrm{T}}\left(\lambda_{k-1}\mathbf{I} + \mathbf{H}_k\mathbf{L}_{k-1}^{-1}\mathbf{H}_k^{\mathrm{T}}\right)^{-1}\mathbf{H}_k\mathbf{L}_{k-1}^{-1}\right)\\ &= \frac{1}{\lambda_{k-1}}\left(\mathbf{P}_{k-1} - \mathbf{P}_{k-1}\mathbf{H}_k^{\mathrm{T}}\left(\lambda_{k-1}\mathbf{I} + \mathbf{H}_k\mathbf{L}_{k-1}^{-1}\mathbf{H}_k^{\mathrm{T}}\right)^{-1}\mathbf{H}_k\mathbf{P}_{k-1}\right).\end{aligned} \tag{5}$$

When $k \geq u$, S_k is acquired, the $(k-u)$-th sample chunk S_{k-u} become invalid. The current valid samples $\overline{S_k} = \bigcup_{i=k-u+1}^{k} S_i$. The relationship between $\boldsymbol{\beta}^{(k)}$ and $\overline{S_k}$ can be expressed as the following linear equation,

$$\begin{bmatrix}\Lambda_{k-u+1}^{(k-1)}\mathbf{H}_{k-u+1}\\ \vdots\\ \Lambda_{k-u+1}^{(k-1)}\mathbf{H}_{k-1}\\ \mathbf{H}_k\end{bmatrix}\beta^{(k)} = \begin{bmatrix}\Lambda_{k-u+1}^{(k-1)}\mathbf{T}_{k-u+1}\\ \vdots\\ \Lambda_{k-u+1}^{(k-1)}\mathbf{T}_{k-1}\\ \mathbf{T}_k\end{bmatrix},$$

where $\Lambda_j^{(k-1)} = \prod_{i=j}^{k-1}\lambda_i, j = k-u+1, ..., k-1$. According to the Moore–Penrose generalized inverse and the minimum norm least square algorithm [47], $\boldsymbol{\beta}^{(k)}$ is denoted,

$$\beta^{(k)} = \begin{bmatrix}\Lambda_{k-u+1}^{(k-1)}\mathbf{H}_{k-u+1}\\ \vdots\\ \Lambda_{k-1}^{(k-1)}\mathbf{H}_{k-1}\\ \mathrm{H}_k\end{bmatrix}^{\dagger}\begin{bmatrix}\Lambda_{k-u+1}^{(k-1)}\mathbf{T}_{k-u+1}\\ \vdots\\ \Lambda_{k-1}^{(k-1)}\mathbf{T}_{k-1}\\ \mathbf{T}_k\end{bmatrix}, \tag{6}$$

where

$$\begin{bmatrix}\Lambda_{k-u+1}^{(k-1)}\mathbf{H}_{k-u+1}\\ \vdots\\ \Lambda_{k-1}^{(k-1)}\mathbf{H}_{k-1}\\ \mathbf{H}_k\end{bmatrix}^{\dagger} = \left(\begin{bmatrix}\Lambda_{k-u+1}^{(k-1)}\mathbf{H}_{k-u+1}\\ \vdots\\ \Lambda_{k-1}^{(k-1)}\mathbf{H}_{k-1}\\ \mathbf{H}_k\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\Lambda_{k-u+1}^{(k-1)}\mathbf{H}_{k-u+1}\\ \vdots\\ \Lambda_{k-1}^{(k-1)}\mathrm{H}_{k-1}\\ \mathbf{H}_k\end{bmatrix}\right)^{-1}\begin{bmatrix}\Lambda_{k-u+1}^{(k-1)}\mathbf{H}_{k-u-1}\\ \vdots\\ \Lambda_{k-1}^{(k-1)}\mathbf{H}_{k-1}\\ \mathbf{H}_k\end{bmatrix}^{\mathrm{T}}.$$

Let $\mathbf{L}_k = (\sum_{i=k-u+1}^{k-1}\Lambda_i^{(k-1)}\mathbf{H}_i^{\mathrm{T}}\mathbf{H}_i + \mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k)$ and $\mathbf{P}_k = \mathbf{L}_k^{-1}$. Now Eq. (6) can be transformed into the following form,

$$\beta^{(k)} = \mathbf{P}_k\begin{bmatrix}\Lambda_{k-u+1}^{(k-1)}\mathbf{H}_{k-u+1}\\ \vdots\\ \Lambda_{k-1}^{(k-1)}\mathbf{H}_{k-1}\\ \mathbf{H}_k\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\Lambda_{k-u+1}^{(k-1)}\mathbf{T}_{k-u+1}\\ \vdots\\ \Lambda_{k-1}^{(k-1)}\mathbf{T}_{k-1}\\ \mathbf{T}_k\end{bmatrix}.$$

The $(k+1)$-th sample chunk S_{k+1} is added into learning. At this time, $\overline{S_{k+1}} = \bigcup_{i=k-u+2}^{k+1} S_i = \overline{S_k} + S_{k+1} - S_{k-s+1}$. The output weight $\beta^{(k+1)}$ takes the following form,

$$\begin{bmatrix}\Lambda_{k-u+2}^{(k)}\mathbf{H}_{k-u+2}\\ \vdots\\ \Lambda_k^{(k)}\mathbf{H}_k\\ \mathbf{H}_{k+1}\end{bmatrix}\beta^{(k+1)} = \begin{bmatrix}\Lambda_{k-u+2}^{(k)}\mathbf{T}_{k-u+2}\\ \vdots\\ \Lambda_k^{(k)}\mathbf{T}_k\\ \mathbf{T}_{k+1}\end{bmatrix},$$

where $\Lambda_j^{(k)} = \prod_{i=j}^{k}\lambda_i, j = k-u+2, \ldots, k$. $\beta^{(k+1)}$ is denoted as follows,

$$\begin{aligned}\beta^{(k+1)} &= \begin{bmatrix}\Lambda_{k-u+2}^{(k)}\mathbf{H}_{k-u+2}\\ \vdots\\ \Lambda_k^{(k)}\mathbf{H}_k\\ \mathbf{H}_{k+1}\end{bmatrix}^{\dagger}\begin{bmatrix}\Lambda_{k-u+2}^{(k)}\mathbf{T}_{k-u+2}\\ \vdots\\ \Lambda_k^{(k)}\mathbf{T}_k\\ \mathbf{T}_{k+1}\end{bmatrix}\\ &= \mathbf{L}_{k+1}^{-1}\begin{bmatrix}\Lambda_{k-u+2}^{(k)}\mathbf{H}_{k-u+2}\\ \vdots\\ \Lambda_k^{(k)}\mathbf{H}_k\\ \mathrm{H}_{k+1}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\Lambda_{k-u+2}^{(k)}\mathbf{T}_{k-u+2}\\ \vdots\\ \Lambda_k^{(k)}\mathbf{T}_k\\ \mathbf{T}_{k+1}\end{bmatrix},\end{aligned} \tag{7}$$

where

$$\begin{aligned}\mathbf{L}_{k+1}&=\left(\sum_{i=k-u+2}^{k}\Lambda_i^{(k)}\mathbf{H}_i^{\mathrm{T}}\mathbf{H}_i+\mathbf{H}_{k+1}^{\mathrm{T}}\mathbf{H}_{k+1}\right)\\&=\lambda_k\mathbf{L}_k+\mathbf{H}_{k+1}^{\mathrm{T}}\mathbf{H}_{k+1}-\Lambda_{k-u+1}^{(k)}\mathbf{H}_{k-u+1}^{\mathrm{T}}\mathbf{H}_{k-u+1}.\end{aligned}$$

$\mathbf{P}_{k+1}$ denotes $\mathbf{L}_{k+1}^{-1}$. According to the Wood-bury formula, and

$$\begin{aligned}\mathbf{P}_{k+1}&=\left(\sum_{i=k-u+2}^{k}\Lambda_i^{(k)}\mathbf{H}_i^{\mathrm{T}}\mathbf{H}_i+\mathbf{H}_{k+1}^{\mathrm{T}}\mathbf{H}_{k+1}\right)^{-1}\\&=\left(\lambda_k\mathbf{L}_k+\mathbf{H}_{k+1}^{\mathrm{T}}\mathbf{H}_{k+1}-\Lambda_{k-u+1}^{(k)}\mathbf{H}_{k-u+1}^{\mathrm{T}}\mathbf{H}_{k-u+1}\right)^{-1}\\&=\left(\lambda_k\mathbf{P}_k^{-1}+\begin{bmatrix}-\Lambda_{k-u+1}^{(k)}\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}\right)^{-1}\\&=\frac{1}{\lambda_k}\left(\mathbf{P}_k-\mathbf{P}_k\begin{bmatrix}-\Lambda_{k-u+1}^{(k)}\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}^{\mathrm{T}}\right.\\&\quad\left.\times\left(\lambda_k\mathbf{I}+\begin{bmatrix}\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}P_k\begin{bmatrix}-\Lambda_{k-u+1}^{(k)}\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}^{\mathrm{T}}\right)^{-1}\begin{bmatrix}\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}\mathbf{P}_k\right).\end{aligned}\tag{8}$$

The right part of Eq. (7) is calculated as follows,

$$\begin{aligned}&\begin{bmatrix}\Lambda_{k-u+2}^{(k)}\mathbf{H}_{k-u+2}\\\vdots\\\Lambda_k^{(k)}\mathbf{H}_k\\\mathbf{H}_{k+1}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\Lambda_{k-u+2}^{(k)}\mathbf{T}_{k-u+2}\\\vdots\\\Lambda_k^{(k)}\mathbf{T}_k\\\mathbf{T}_{k+1}\end{bmatrix}\\&=\left(\sum_{l=k-u+2}^{k}\left(\Lambda_l^{(k)}\right)^2\mathbf{H}_l^{\mathrm{T}}\mathbf{T}_l+\mathbf{H}_{k+1}^{\mathrm{T}}\mathbf{T}_{k+1}\right)\\&=\begin{bmatrix}\Lambda_{k-u+1}^{(k)}\mathbf{H}_{k-u+1}\\\vdots\\\Lambda_{k-1}^{(k)}\mathbf{H}_{k-1}\\\Lambda_k^{(k)}\mathbf{H}_k\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\Lambda_{k-u+1}^{(k)}\mathbf{T}_{k-u+1}\\\vdots\\\Lambda_{k-1}^{(k)}\mathbf{T}_{k-1}\\\mathbf{T}_k\end{bmatrix}+\begin{bmatrix}-\Lambda_{k-u+1}^{(k)}\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}\\&\quad\times\begin{bmatrix}\Lambda_{k-1}^{(k)}\mathbf{T}_{k-u+1}\\\mathbf{T}_{k+1}\end{bmatrix}\\&=\lambda_k^2\mathbf{P}_k^{-1}\mathbf{P}_k\begin{bmatrix}\Lambda_{k-u+1}^{(k-1)}\mathbf{H}_{k-u+1}\\\vdots\\\Lambda_{k-1}^{(k-1)}\mathbf{H}_{k-1}\\\mathbf{H}_k\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\Lambda_{k-u+1}^{(k-1)}\mathbf{T}_{k-u+1}\\\vdots\\\Lambda_{k-u+1}^{(k-1)}\mathbf{T}_{k-1}\\\mathbf{T}_k\end{bmatrix}\\&\quad+\begin{bmatrix}-\Lambda_{k-u+1}^{(k)}\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}-\Lambda_{k-u+1}^{(k)}\mathbf{T}_{k-u+1}\\\mathbf{T}_{k+1}\end{bmatrix}\\&=\lambda_k^2\mathbf{P}_k^{-1}\beta^{(k)}+\begin{bmatrix}-\left(\Lambda_{k-u+1}^{(k)}\right)^2\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{T}_{k-u+1}\\\mathbf{T}_{k+1}\end{bmatrix}.\end{aligned}$$

Then, Eq. (7) can be transformed into the following,

$$\begin{aligned}\beta^{(k+1)}&=\mathbf{P}_{k+1}\left(\lambda_k^2\mathbf{P}_k^{-1}\beta^{(k)}+\begin{bmatrix}-\left(\Lambda_{k-u+1}^{(k)}\right)^2\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{T}_{k-u+1}\\\mathbf{T}_{k+1}\end{bmatrix}\right)\\&=\mathbf{P}_{k+1}\left(\begin{aligned}&\lambda_k^2\left(\mathbf{L}_{k+1}-\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k+\Lambda_{k-u+1}^{k}\mathbf{H}_{k-u+1}^{\mathrm{T}}\mathbf{H}_{k-u+1}\right)\beta^{(k)}\\&+\begin{bmatrix}-\Lambda_{k-u+1}^{(k)}\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\Lambda_{k-u+1}^{(k)}\mathbf{T}_{k-u+1}\\\mathbf{T}_{k+1}\end{bmatrix}\end{aligned}\right)\\&=\mathbf{P}_{k+1}\left(\begin{aligned}&\lambda_k^2\left(\mathbf{P}_{k+1}^{-1}-\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k+\Lambda_{k-u+1}^{k}\mathbf{H}_{k-u+1}^{\mathrm{T}}\mathbf{H}_{k-u+1}\right)\beta^{(k)}\\&+\begin{bmatrix}-\Lambda_{k-u+1}^{(k)}\mathbf{H}_{k-s+1}\\\mathbf{H}_{k+1}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\Lambda_{k-u+1}^{(k)}\mathbf{T}_{k-u+1}\\\mathbf{T}_{k+1}\end{bmatrix}\end{aligned}\right)\\&=\mathbf{P}_{k+1}\left(\begin{aligned}&\lambda_k^2\left(\mathbf{P}_{k+1}^{-1}-\begin{bmatrix}-\Lambda_{k-u+1}^{(k)}\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathrm{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}\right)\beta^{(k)}\\&+\begin{bmatrix}-\Lambda_{k-u+1}^{(k)}\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\Lambda_{k-u+1}^{(k)}\mathrm{T}_{k-u+1}\\\mathbf{T}_{k+1}\end{bmatrix}\end{aligned}\right)\\&=\lambda_k^2\beta^{(k)}+\mathbf{P}_{k+1}\begin{bmatrix}-\Lambda_{k-u+1}^{(k)}\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}^{\mathrm{T}}\left(\begin{bmatrix}\Lambda_{k-u+1}^{(k)}\mathbf{T}_{k-u+1}\\\mathbf{T}_{k+1}\end{bmatrix}-\lambda_k^2\begin{bmatrix}\mathbf{H}_{k-u+1}\\\mathbf{H}_{k+1}\end{bmatrix}\beta^{(k)}\right).\end{aligned}\tag{9}$$

3.2. The memory factor

To adjust the weights of the old and new samples in a MDOS-ELM model, the memory factor λ, $(0<\lambda<1)$ is employed. When λ is a constant value, the dynamic characteristics of the samples are not sensitively reflected. When λ is given by experts, the accuracy of $\boldsymbol{\beta}$ is usually bad affected. In order to achieve better performance of a MDOS-ELM model, the memory factor with the ability of updating is required. λ is related to two elements. One is the similarity between the new and old samples, and the other is the prediction errors of the current training samples on the previous model.

The exponential function is employed to compute the memory factor, which can guarantee that the memory factor ranges from 0 to 1. The memory factor can be expressed as

$$\lambda_k=1-\exp\left(-\frac{\Upsilon_k}{\xi_k}\right),\ k=0,1,2,\dots,\infty,\tag{10}$$

where Υ_k denotes the similarity between the kth and the (k+1)-th sample chunks, and ξ_k denotes the prediction errors of the current samples from the previous model. Because $-\frac{\Upsilon_k}{\xi_k}<0$, the right side of Eq. (10) belong 0 and 1.

If λ_k equals one, the difference between the $(k+1)$th and the k-th sample chunks does not exist. If λ_k equals zero, the proposed MDOS-ELM degenerates to the traditional ELM that is built by the $(k+1)$th chunk samples. If λ_k identically equals one, the MDOS-ELM becomes the FOS-ELM. If both $u=+\infty$ and the λ_k identically equals one, the MDOS-ELM changes to the OS-ELM. Therefore, the FOS-ELM and the OS-ELM are special cases of the MDOS-ELM.

Fig. 1 is the diagram of the MDOS-ELM model sequence. When the $(k+1)$th sample chunk S_{k+1} is acquired, the MDOS-ELM model is updated and then Model$_{k+1}$ is obtained. At the moment, memories of Model$_{k+1}$ for S_k and S_{k-1} are λ_k and $\lambda_k\cdot\lambda_{k-1}$, respectively. Memories of Model$_{k+1}$ for S_j can be analogized as $\prod_{l=j}^{k}\lambda_l$, $j=k-u+1,\dots,k-1,k$. The stronger the similarity between S_k and S_{k+1} is, the closer contributions of S_k and S_{k+1} to Model$_{k+1}$ is, and then the larger the λ_k is [48].

Algorithm 3
The proposed MDOS-ELM algorithm.

```
Phase I Initialization
As mentioned above, $S_0$ is an initial sample chunk, $L$ is the number of the hidden layer nodes, $G(\cdot)$ is the
activation function, $u$ is the period of validity. And similarity calculation method is selected.
Step 1, parameters of hidden layer $\mathbf{a}$ and $b$ are randomly assigned.
Step 2, the hidden layer output matrix $\mathbf{H}_0$ is computed.
Step 3, the initial output weights $\beta^{(0)} = \mathbf{P}_0\mathbf{H}_0^{\mathrm{T}}\mathbf{T}_0$ is computed.
Phase II Online sequential learning
The $(k+1)-$th sample chunk $S_{k+1}$ is added into learning.
Step 4, according to Eq. (10) the memory factor $\lambda_k$ is computed.
Step 5, the output matrix $\mathbf{H}_{k+1}$ of the hidden layer is computed.
Step 6, the output weight $\beta^{(k+1)}$ is computed.
    When $k+1 < u$, according to Eq. (4) $\beta^{(k+1)}$ is computed.
    When $k+1 \geq u$, according to Eq. (9) $\beta^{(k+1)}$ is computed.
    Then, the current SLFN model can be obtained based on Eq. (1).
Step 7, when a new sample chunk is added into learning, $k$ is replaced by $k+1$ and then turn to Step 4.
```

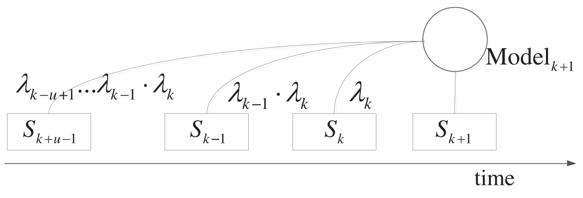

Model$_{k+1}$
$\lambda_{k-u+1}\cdots\lambda_{k-1}\cdot\lambda_k$
$\lambda_{k-1}\cdot\lambda_k$
λ_k
S_{k+u-1}
S_{k-1}
S_k
S_{k+1}
time

Fig. 1. The diagram of the MDOS-ELM model sequence.

$\overline{\mathbf{x}}^{(k)} = [\overline{x}_1{}^{(k)}, \overline{x}_2{}^{(k)}, \ldots, \overline{x}_n{}^{(k)}]$ denotes the average of the input matrix in the kth sample chunk, where $\overline{x}_i{}^{(k)}$ is the average of the ith input variable. Here, the Euclidean distance, the Cosine, the Pearson-Correlation coefficient or the Robust-Huber Similarity Measure (RHSM) [44] can be applied to compute the similarity of sample chunk.

① The similarity based on the Euclidean distance is

$$\Upsilon_k = \frac{1}{1 + \left\| \overline{\mathbf{x}}^{(k)} - \overline{\mathbf{x}}^{(k+1)} \right\|_2}, \tag{11}$$

where $\|\overline{\mathbf{x}}^{(k)} - \overline{\mathbf{x}}^{(k+1)}\|_2$ represents the Euclidean distance between $\overline{\mathbf{x}}^{(k)}$ and $\overline{\mathbf{x}}^{(k+1)}$.

② The similarity based on Cosine is

$$\Upsilon_k = \frac{\cos\left(\overline{\mathbf{x}}^{(k)}, \overline{\mathbf{x}}^{(k+1)}\right)}{2} + \frac{1}{2}, \tag{12}$$

where$\cos(\overline{\mathbf{x}}^{(k)}, \overline{\mathbf{x}}^{(k+1)}) = \frac{\langle \overline{\mathbf{x}}^{(k)}, \overline{\mathbf{x}}^{(k+1)} \rangle}{\|\overline{\mathbf{x}}^{(k)}\|_2 \|\overline{\mathbf{x}}^{(k+1)}\|_2}$,and $\langle \overline{\mathbf{x}}^{(k)}, \overline{\mathbf{x}}^{(k+1)} \rangle$ denotes the inner product between $\overline{\mathbf{x}}^{(k)}$ and $\overline{\mathbf{x}}^{(k+1)}$.

③ The similarity based on Pearson-Correlation is

$$\Upsilon_k = \frac{p\left(\overline{\mathbf{x}}^{(k)}, \overline{\mathbf{x}}^{(k+1)}\right)}{2} + \frac{1}{2}, \tag{13}$$

where $p(\overline{\mathbf{x}}^{(k)}, \overline{\mathbf{x}}^{(k+1)})$ represents the Pearson-Correlation coefficient between $\overline{\mathbf{x}}^{(k)}$ and $\overline{\mathbf{x}}^{(k+1)}$,and $p(\overline{\mathbf{x}}^{(k)}, \overline{\mathbf{x}}^{(k+1)}) = \frac{\mathrm{E}(\overline{\mathbf{x}}^{(k)}\overline{\mathbf{x}}^{(k+1)}) - \mathrm{E}(\overline{\mathbf{x}}^{(k)})\mathrm{E}(\overline{\mathbf{x}}^{(k+1)})}{\sqrt{\mathrm{D}(\overline{\mathbf{x}}^{(k)})}\sqrt{\mathrm{D}(\overline{\mathbf{x}}^{(k+1)})}}$. $\mathrm{E}(\overline{\mathbf{x}}^{(k)})$ is the mathematical expectation of variable $\overline{\mathbf{x}}^{(k)}$ and $\mathrm{D}(\overline{\mathbf{x}}^{(k)})$ is the variance of variable $\overline{\mathbf{x}}^{(k)}$.

④ The similarity based on RHSM is [44]

$$\Upsilon_k = \frac{1}{1 + \mathrm{RHSM}\left(\overline{\mathbf{x}}^{(k)}, \overline{\mathbf{x}}^{(k+1)}\right)}, \tag{14}$$

where

$$\mathrm{RHSM}\left(\overline{\mathbf{x}}^{(k)}, \overline{\mathbf{x}}^{(k+1)}\right) = \sum_{i=1}^{n} \rho_{\mathrm{H}}\left(\overline{x_i}^{(k)} - \overline{x_i}^{(k+1)}\right).$$

$\rho_{\mathrm{H}}(r)$ is the Huber function [45], and

$$\rho_{\mathrm{H}}(r) = \begin{cases} \frac{1}{2}r^2, & |r| \leq \alpha \\ \alpha|r| - \frac{1}{2}\alpha^2, & \text{otherwise} \end{cases},$$

where $\alpha(\alpha > 0)$ denotes the contamination degree of outliers. And it referred to as the parameter of the Huber function. Aforementioned four methods can guarantee similarity that belongs 0 to 1.

In addition, λ is also related to the prediction error of the previous model on current samples. $\mathbf{H}_{k+1}\beta^{(k)}$ denotes the predictions of the current samples by the previous model. $\xi_k = ||\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\beta^{(k)}||_2$ denotes the prediction error of the current samples from the previous model. The smaller the ξ_k is, the more accurate of the previous model on the current samples is, and the larger contribution of the previous model to the current model. The MDOS-ELM algorithm is given in Algorithm 3.

4. Experiments

In order to verify the performance of the proposed MDOS-ELM, nine datasets of regression and fourteen datasets of classification were employed to carry out the experiments. These datasets included an artificial dataset and twenty-two real-world datasets. These real-world datasets were collected form the UCI machine learning repository, the Delve Datasets and the StatLib-Datasets Archive. Twonorn was collected form the Delve Datasets and Bodyfat was collect form the StatLib—Datasets Archive. The rest real-world datasets came from the UCI machine learning repository. Their essential information was given in Table 1. Additionally, two related online ELM versions (the OS-ELM and the FOS-ELM) were treated as comparisons with the proposed MDOS-ELM.

In all the runs the following procedures were used.

① All continuous variables were normalized as follows,

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}},$$

$$y^* = \frac{y - y_{\min}}{y_{\max} - y_{\min}}.$$

For the regression datasets, all the input and output variables were normalized into the [0,1] interval. And for the classification datasets, the input variables also were normalized.

② For the regression datasets, the Root Mean Square Error (RMSE) was used as the evaluative criteria for the accuracy and generalization. The Coefficient of Determination R^2 measured the stability of the predictive model. $\hat{y}$ denoted the es-

Table 1
Essential information of real-world datasets.

Dataset	Task type	Classes	Instances		Input features
			Training	Testing	
Magic04	Classification	2	15,216	3804	10
Twonorn	Classification	2	5920	1480	20
Pendigits	Regression	–	8794	2098	16
Letter	Regression	–	16,000	4000	16
Breast-Tissue	Classification	6	84	22	9
Glass	Classification	6	163	51	9
Bodyfat	Regression	–	202	50	14
Triazines	Regression	–	146	40	60
Auto-MPG	Regression	–	320	72	7
Abalone	Regression	–	3000	1177	8
California Housing	Regression	–	8000	12,640	8
Zoo	Classification	7	71	30	17
Wine	Classification	3	120	58	13
New-Thyroid	Classification	3	140	75	5
Monks-1	Classification	2	300	132	6
Image segmentation	Classification	7	1500	810	19
Satellite image	Classification	6	4435	2000	26
Mackey-Glass	Regression	–	4000	500	4
Vehiche	Classification	4	630	216	18
Adult	Classification	2	40,699	4523	14
Optical digits	Classification	10	1200	4420	64
Mushroom	Classification	2	7311	813	21

timated value, and y the real value.

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{j=1}^{N}\left(\hat{y}_i - y_i\right)}{N}},$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}\left(\hat{y}_i - y_i\right)^2}{\sum_{i=1}^{N}\left(y_i - \bar{y}\right)^2}.$$

The Coefficient of Determination R^2 is also called goodness of fit. Its range interval is $\in[0, 1]$. The bigger R^2 is, the better the prediction model is. That is, the generalization of the prediction model is better.

③ For the classification datasets, the correct rate measured the performance of the algorithm.

$$Accuracy = \frac{N^*}{N}.$$

N samples were predicted, and N^* samples were correctly predicted.

④ All the activation functions of the OS-ELM, the FOS-ELM and the MDOS-ELM adopt “Sigmoid” function.

$$g(\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{a}\cdot\mathbf{x}+b)}}.$$

⑤ All the experimental results were statistical. Namely the same experiment was performed 50 times, and then the average value was obtained.

4.1. The experiment on the regression datasets

In order to verify the performance of the DMOS-ELM model on the regression problem, an artificial dataset (a standard test function, Friedman#2) and eight regression datasets from the real world were employed.

4.1.1. The experiment on an artificial dataset

Standard test function Friedman#2

$$y = x_1^2 + \sqrt{\frac{x_2x_3 - 1}{(x_2x_4)^2}},$$

Table 2
The RMSE and R^2 of MDOS-ELM models with three similarity methods on lower noises Friedman#2.

Methods of similarity measurement	Training		Testing	
	RMSE	R^2	RMSE	R^2
Euclidean distance	0.0136	0.9981	0.0196	0.9962
Cosine	0.0147	0.9922	0.0216	0.9924
Pearson-Correlation coefficient	0.0152	0.9912	0.0281	0.9916

Table 3
The RMSE and R^2 of MDOS-ELM models with the RHSM on lower noises Friedman#2.

α	Training		Testing	
	RMSE	R^2	RMSE	R^2
0.01	0.0171	0.9912	0.0196	0.9882
0.02	0.0133	0.9978	0.0182	0.9908
0.1	0.0131	0.9947	0.0194	0.9921
0.5	0.0123	0.9984	0.0186	0.9941
1	0.0127	0.9918	0.0198	0.9868
2	0.0126	0.9931	0.0191	0.9831
3	0.0126	0.9912	0.0197	0.9832

where $x_1 \in [0, 99]$, $x_2 \in [4\pi, 564\pi - 1]$, $x_3 \in [0, 1]$ and $x_4 \in [1, 11]$. 2000 training samples and 1000 testing samples were randomly generated. The 2000 training samples were divided into 19 sample chunks (an initial sample chunk and 18 online updating-sample chunks). The initial sample chunk contained 200 samples, and each online updating-sample chunk contained 100 samples. The number of the hidden layer nodes of the OS-ELM, the FOS-ELM and the MDOS-ELM were all set as N_0. In order to test the robustness of the proposed MDOS-ELM, lower and higher white noises were added into the artificial dataset, respectively.

① Add lower white noises

The random white noises distributed normally $N\sim(0, 3)$ were added into all input variables the training samples. If the period of validity (i.e.u) of the FOS-ELM and the MDOS-ELM were set too large, the model covered the outdated information of samples. On the contrary, the model did not cover the information of the near samples. The optimal u cannot be known in advance, and there is no good method to determine it in theory. Common method is constantly trial. Both the FOS-ELM model and the MDOS-ELM model with different u respectively were experimented. The results were shown in Fig. 2(a) and (b), respectively. Here, the Euclidean distance was selected to compute the similarity for the MDOS-ELM.

Fig. 2(a) and (b) showed that both u period of the validity of the FOS-ELM and that of the MDOS-ELM was 5 unit time ($u = 5$), which could obtain the lowest RMSE on the testing set. Thus, both u of the MDOS-ELM and that of the FOS-ELM were 5 unit time.

Additionally, the experiment analyzed that the memory factor was calculated by different similarity methods to impact on the performance of the MDOS-ELM. The experimental results of the MDOS-ELM models with common similarity methods (the Euclidean distance, the Cosine, the Pearson-Correlation coefficient) were shown in Table 2. Table 2 showed that all the MDOS-ELM with the different similarity methods could get the expected results. It was best to use the Euclidean distance method expressed similarity in three common methods. The MDOS-ELM models with different RHSM computed similarity were experimented, and then results were shown in Table 3. These results displayed little difference between them. The reason was that noises in the dataset were relatively small.

The OS-ELM and the FOS-ELM, related online ELM version, were employed for the comparisons with the proposed MDOS-ELM. The comparison results were displayed in Table 4. In the first row of

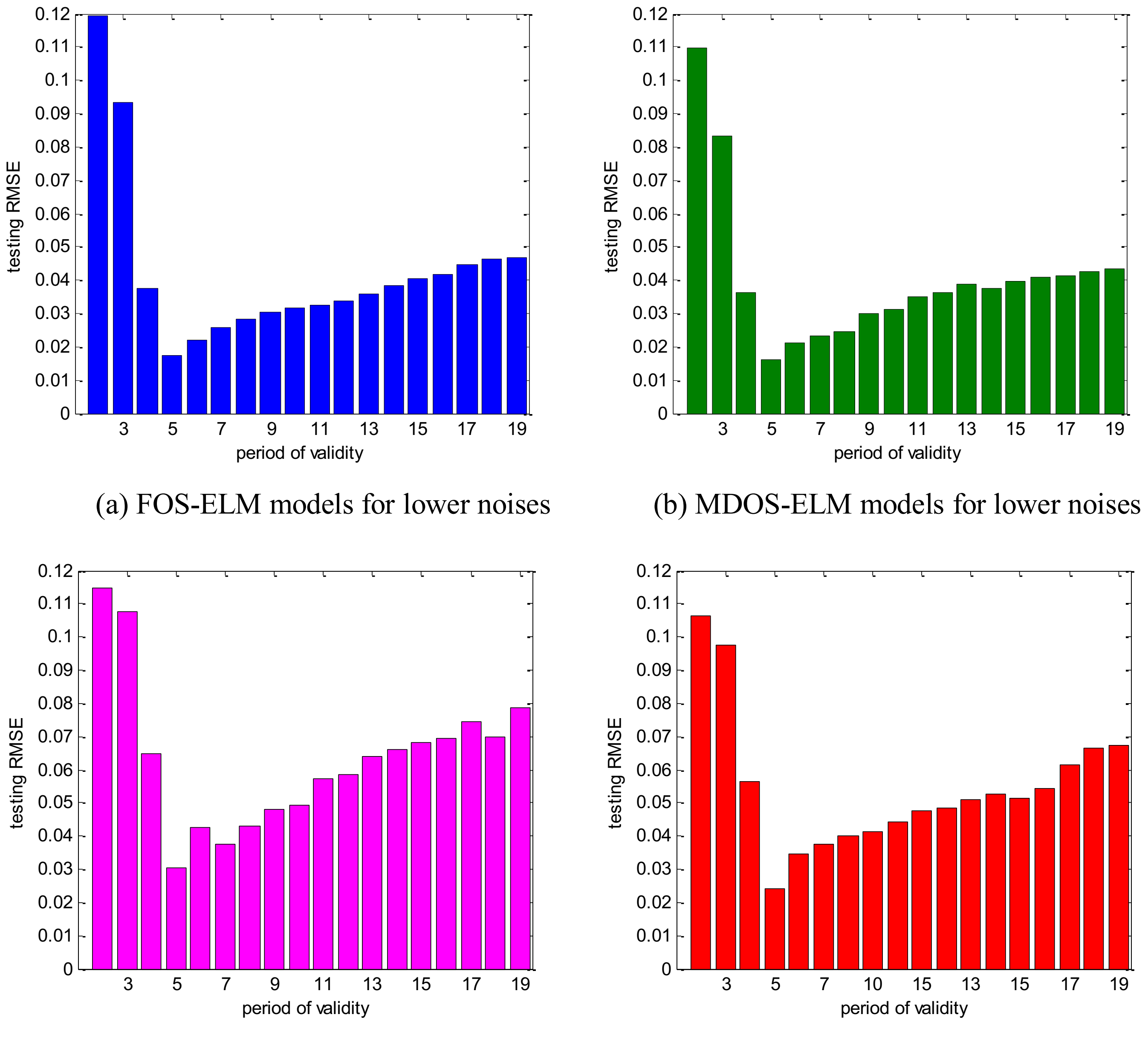


Fig. 2. Comparisons of FOS-ELM and MDOS-ELM models with different u on the Fredman#2.

Table 4
The RMSE and R^2 of the three kinds of model on lower noises Friedman#2.

	Methods of similarity measurement	Training		Testing	
		RMSE	R^2	RMSE	R^2
MDOS-ELM	Euclidean distance	0.0136	0.9981	0.0196	0.9962
	RHSM	0.0123	0.9984	0.0186	0.9961
OS-ELM	–	0.0233	0.9778	0.0281	0.9842
FOS-ELM	–	0.0204	0.9878	0.0254	0.9832

Table 4, this similarity method was Euclidean distance. The Huber function parameter of RHSM α was set as 0.5 in the second row of Table 4.

Table 4 results showed that the RMSE of the MDOS-ELM model was smaller than that of the OS-ELM and the OS-ELM. The R^2 of the MDOS-ELM is larger than that of the OS-ELM and the OS-ELM. It demonstrated that the generalization of the prediction model built by the MDOS-ELM was better than the prediction models built by the OS-ELM and the OS-ELM.

② Add higher white noises

In odder to further verify the robustness of the proposed MDOS-ELM and contrast the robustness of three kinds of model, noises distributed normally $N\sim(0, 12)$ were added into all the input variables the training samples. The FOS-ELM and MDOS-ELM

Table 5
The RMSE and $R^2$of MDOS-ELM models with three common similarity methods on higher-noise Friedman#2.

Methods of similarity measurement	Training		Testing	
	RMSE	R^2	RMSE	R^2
Euclidean distance	0.0267	0.9801	0.0325	0.9783
Cosine	0.0311	0.9701	0.0402	0.9614
Pearson-Correlation coefficient	0.0282	0.9721	0.0281	0.9626

Table 6
The RMSE and R^2 of MDOS-ELM models with the RHSM on higher noises Friedman#2.

α	Training		Testing	
	RMSE	R^2	RMSE	R^2
0.01	0.0311	0.9712	0.0489	0.9782
0.02	0.0293	0.9733	0.0382	0.9701
0.1	0.0204	0.9792	0.0345	0.9634
0.5	0.0176	0.9931	0.0216	0.9812
1	0.0212	0.9689	0.0482	0.9318
2	0.0322	0.9611	0.0452	0.9601
3	0.0316	0.9631	0.0451	0.9612
5	0.0321	0.9719	0.0459	0.9622

models with different u were experimented, and then the results were shown in Fig. 2(c) and (d), respectively. The lowest testing RMSE were obtained by the FOS-ELM and the MDOS-ELM models, when u were set as 5. Thus, both the u of the MDOS-ELM and that of the FOS-ELM were set as 5.

The experimental results of the MDOS-ELM with memory factor calculated by the three common similarity methods were shown in Table 5. Table 5 results showed that similarly computed by Euclidean distance was better than others. For memory factor calculated by the RHSM, the parameter of Huber function α was set as different values to carry out the experiment, the results of which were shown in Table 6.

The experimental effect is better, when the parameter of Huber function α was set as 0.5. When $\alpha > 1$, the RMSE of the MDOS-ELM were bigger. The reason of outlier makings [42] (false negative detections—not detected outliers) might happen. When α was set too small, these experimental results was still not better. The reason is outlier swamping [42] (false positive detections—correct values treated as outliers) might appear. Table 7 presented the comparison results of three kinds of model. It showed that the results of the MDOS-ELM were better than these of the OS-ELM and FOS-ELM.

Lower and higher noises were added into the training samples, respectively. The RMSE of MDOS-ELM were smaller than that of the OS-ELM and the FOSELM. Moreover, the R^2 of the MDOS-ELM was greater than that of the OS-ELM and OS-ELM. The vibration of the prediction of the MDOS-ELM was slightly smaller than that of the OS-ELM and the FOS-ELM. When samples appear to be disturbed, the predictions obtained by MDOS-ELM were relatively stable [49]. Thus, the robustness of the MDOS-ELM was better than that of the OS-ELM and the FOS-ELM.

4.1.2. The experiments on the real-world regression datasets

In order to further verify the performance of the MDOS-ELM dealing with the regression problem, eight real-world regression datasets were employed. In this section, Pendigits, Letter, Bodyfat and Triazines were analyzed in detail. The Pendigits dataset and Letter dataset were big datasets (Pendigits and Letter respectively included 10,992 samples and 20,000). The training samples of Pendigits were divided into an initial sample chunk including 2000 samples and 68 online updating-sample chunks. Each online updating-sample chunk contained 100 samples. The training samples of Letter dataset were divided into an initial sample chunk including 2000 samples and 140 online updating-sample chunks. Form Fig. 3(a) and (b), when the u of the FOS-ELM and that of the MDOS-ELM models were 12, the lowest testing RMSE were obtained for Pendigits datasets. So both the u of the FOS-ELM and that of the MDOS-ELM were set as 12. The same method was used to set u of other regression datasets. The u of the MDOS-ELM model and that of FOS-ELM model were set as 10, 5 and 7 for Letter, Bodyfat and Triazines, respectively.

Table 8 displayed the results of the MDOS-ELM with memory factors computed by the three common similarity methods. For memory factors were computed by RHSM, the parameter of the RHSM was set as different values to carry out the experiment, results of which were presented in Table 9. Table 8 showed that the Cosine and the Pearson-Correlation coefficient were better than others on the Pendigits dataset and Letter dataset, respectively. Table 9 demonstrated that $\alpha = 0.5$ and $\alpha = 1$ results were better than others on the Pendigits dataset and Letter dataset, respectively.

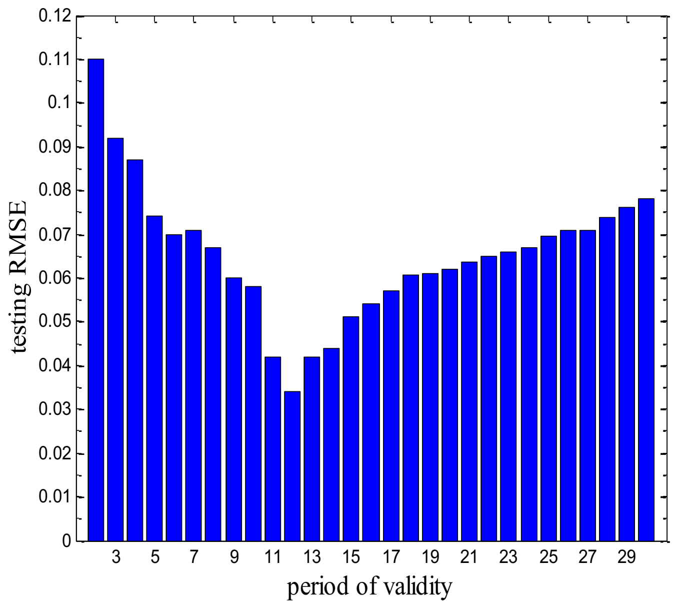
Table 10 presented the comparison results of the three kinds of model. It showed the RMSE of the MDOS-ELM were smaller than that of the other two kinds of model, and the R^2 of the MDOS-ELM were larger than that of the two kinds of model. They indicated that the prediction model established by the MDOS-ELM was better than the other two kinds of model. Here, the parameter of Huber function of MDOS-ELM was $\alpha = 0.5$ and $\alpha = 1$ for the Pendigits and Letter datasets, respectively. The RMSE and R^2 of the MDOS-ELM that the RHSM computed similarly were slightly better than these of the MDOS-ELM that computed similarly by common methods.

Triazines and Bodyfat contained 186 samples and 214 samples, respectively. Triazines dataset was a high-dimensional dataset with a total of 60 input features. The training samples of Triazines were divided into 14 sample chunks (an initial sample chunk and 13 an online updating sample chunk). Among these sample chunks, an initial sample chunk contained 20 samples. Each online updating sample chunk included 10 samples. The training samples of the Bodyfat dataset were divided into an initial sample chunk and 16 online updating sample chunks. The initial sample chunk included 50 samples. Each online updating samples chunk contained 10 samples.

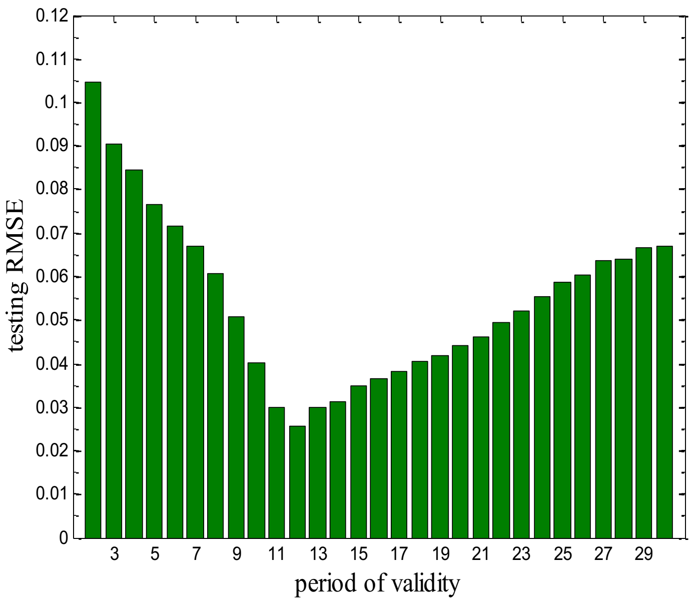
The experimental results of the MDOS-ELM with memory factor calculated by the three common similarity methods were given in Table 11, which showed that the similarity computed by Pearson-Correlation coefficient and Cosine was better than that by the Tri-

Table 7
The RMSE and R^2 of the three kinds of model on higher noises Friedman#2.

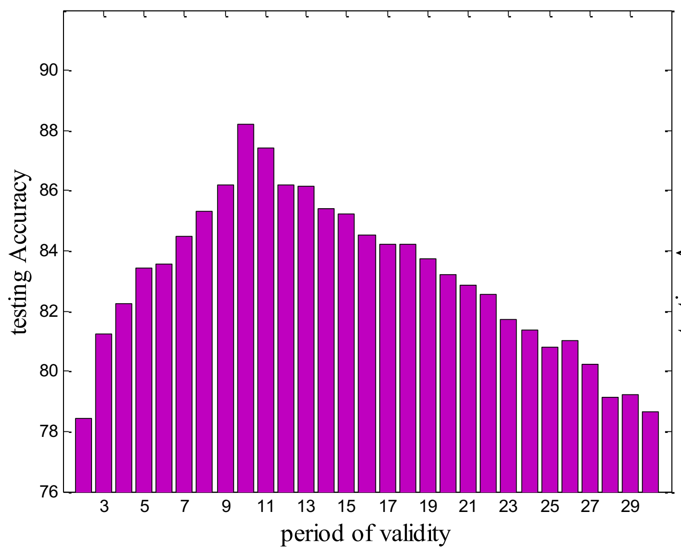
Algorithm	Methods of similarity measurement	Training		Testing	
		RMSE	R^2	RMSE	R^2
MDOS-ELM	Euclidean distance	0.0267	0.9801	0.0325	0.9783
	RHSM	0.0176	0.9931	0.0216	0.9812
OS-ELM	–	0.0405	0.9678	0.0439	0.9592
FOS-ELM	–	0.0380	0.9718	0.0394	0.9836



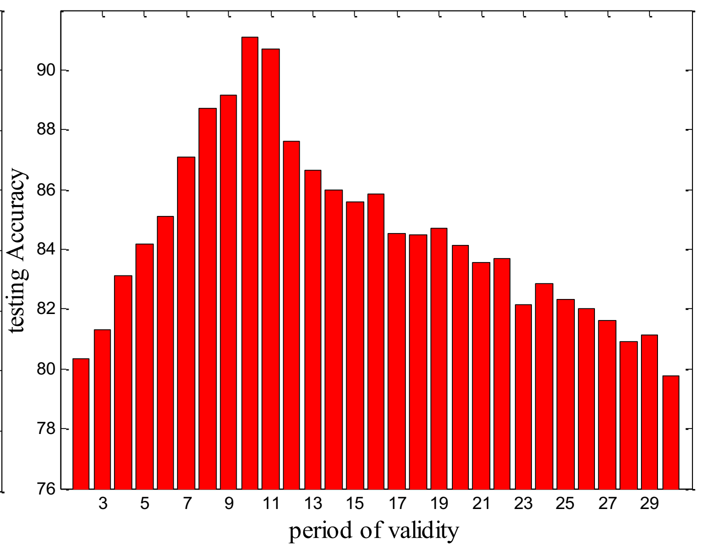
(a) FOS-ELM models for Pendigits



(b) MDOS-ELM models for Pendigits



(c) FOS-ELM models for Magic04



(d) MDOS-ELM models for Magic04

Fig. 3. Comparisons of FOS-ELM and MDOS-ELM models with different u on Pendigits and Magic04.

Table 8
The RMSE and R^2 of MDOS-ELM models with three commonly similarity methods on the Pendigits and Letter datasets.

Methods of similarity measurement	Pendigits				Letter			
	Training		Testing		Training		Testing	
	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2
Euclidean distance	0.0434	0.9721	0.0514	0.9544	0.0404	0.9763	0.0584	0.9633
Cosine	0.0481	0.9722	0.0514	0.9521	0.0472	0.9782	0.0504	0.9642
Person-Correlation-Coefficient	0.0324	0.9772	0.0523	0.9573	0.0396	0.9792	0.0462	0.9735

azines dataset and Bodyfat dataset, respectively. For memory factor calculated by the RHSM, the parameter of Huber function was set as different values to carry out the experiment, results of which were displayed in Table 12. When $\alpha = 0.5$, results were better on these two datasets.

Table 13 presented the comparison results of three kinds of model. Table 13 showed the RMSE of MDOS-ELM was smaller than that of the other two kinds of model. And the R^2 of the MDOS-ELM was also larger than that of two kinds of models. They indicated that the generalization ability and stability of the prediction model

Table 9
The RMSE and R^2 of MDOS-ELM models with different the RHSM on the Pendigits and Letter datasets.

α	Pendigits				Letter			
	Training		Testing		Training		Testing	
	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2
0.01	0.0617	0.9613	0.0765	0.9312	0.0773	0.9464	0.0814	0.9311
0.02	0.0529	0.9817	0.1061	0.8877	0.0485	0.9382	0.0496	0.9221
0.1	0.0497	0.9491	0.0585	0.9264	0.0513	0.9819	0.0548	0.9526
0.5	0.0319	0.9813	0.0446	0.9876	0.0582	0.9776	0.0600	0.9621
1	0.0412	0.9821	0.0819	0.9636	0.0346	0.9902	0.0373	0.9836
2	0.0565	0.9712	0.0756	0.9466	0.0410	0.9603	0.0445	0.9532
3	0.0535	0.9731	0.0599	0.9568	0.0419	0.9689	0.0481	0.9536
5	0.0515	0.9747	0.0825	0.9087	0.0412	0.9709	0.0419	0.9524

Table 10
The RMSE and R^2 of the three kinds of model on the Pendigits and Letter datasets.

Datasets	Algorithm	Methods of similarity measurement	Training		Testing	
			RMSE	R^2	RMSE	R^2
Pendigits	MDOS-ELM	Cosine	0.0324	0.9672	0.0543	0.9573
		RHSM	0.0319	0.9813	0.0446	0.9876
	OS-ELM	–	0.0767	0.8731	0.0827	0.8979
	FOS-ELM	–	0.0473	0.9723	0.0517	0.9312
Letter	MDOS-ELM	Pearson-Correlation coefficient	0.0396	0.9792	0.0462	0.9735
		RHSM	0.0346	0.9902	0.0373	0.9836
	OS-ELM	–	0.0677	0.8781	0.0727	0.8973
	FOS-ELM	–	0.0517	0.9672	0.0673	0.9423

Table 11
The RMSE and R^2 of MDOS-ELM models with three common similarity methods on the Triazines and Bodyfat datasets.

Methods of similarity measurement	Triazines				Bodyfat			
	Training		Testing		Training		Testing	
	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2
Euclidean distance	0.0951	0.8521	0.0906	0.9094	0.0815	0.9023	0.0861	0.9114
Cosine	0.0883	0.8922	0.0916	0.9022	0.0613	0.9422	0.0706	0.9424
Pearson-Correlation coefficient	0.0783	0.9372	0.0885	0.9052	0.0748	0.9372	0.0801	0.9212

Table 12
The RMSE and R^2 of MDOS-ELM models with different the RHSM on the Triazines and Bodyfat datasets.

α	Triazines				Bodyfat			
	Training		Testing		Training		Testing	
	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2
0.01	0.0936	0.8981	0.0916	0.8901	0.0815	0.9023	0.0968	0.9192
0.02	0.0929	0.8911	0.0826	0.8811	0.0813	0.9122	0.0969	0.9426
0.1	0.0849	0.9355	0.0885	0.9135	0.0748	0.9273	0.0889	0.9014
0.5	0.0747	0.9605	0.0806	0.9025	0.0615	0.9433	0.0735	0.9432
1	0.0916	0.9243	0.0916	0.9143	0.0695	0.9422	0.0826	0.9031
2	0.1002	0.8627	0.1036	0.8827	0.0751	0.932	0.0892	0.9233
3	0.0930	0.9187	0.1006	0.8946	0.0731	0.9342	0.0869	0.9213
5	0.0893	0.9017	0.0875	0.8917	0.0748	0.9372	0.0823	0.9134

Table 13
The RMSE and R^2 of the three kinds of model on the Triazines and Bodyfat datasets.

Datasets	Algorithm	Methods of similarity measurement	Training		Testing	
			RMSE	R^2	RMSE	R^2
Pendigits	MDOS-ELM	Pearson-Correlation coefficient	0.0783	0.9372	0.0885	0.9052
		RHSM	0.0647	0.9605	0.0706	0.9325
	OS-ELM	–	0.1113	0.8912	0.1176	0.8734
	FOS-ELM	–	0.0969	0.8921	0.0982	0.8972
Letter	MDOS-ELM	Pearson-Correlation coefficient	0.0613	0.9422	0.0706	0.9424
		RHSM	0.0515	0.9433	0.0505	0.9432
	OS-ELM	–	0.0901	0.9021	0.0976	0.8934
	FOS-ELM	–	0.0843	0.9342	0.0892	0.9023

Table 14
Experimental results of the three kinds of model on other regression datasets.

Datasets	Algorithms	Methods of similarity measurement	u	α	RMSE	
					Training	Testing
Auto-MPG	MDOS-ELM	Pearson-Correlation coefficient	10	–	0.0632	0.0693
		RHSM	10	0.5	0.0628	0.0673
	OS-ELM	–	–	–	0.0682	0.0742
	FOS-ELM	–	10	–	0.0642	0.0763
Abalone	MDOS-ELM	Person-Correlation-coefficient	12	–	0.0746	0.0753
		RHSM	12	2	0.0725	0.0764
	OS-ELM	–	–	–	0.0776	0.0789
	FOS-ELM	–	12	–	0.0764	0.0772
California Housing	MDOS-ELM	Cosine	10	–	0.0846	0.0861
		RHSM	10	0.5	0.0814	0.0843
	OS-ELM	–	–	–	0.1068	0.1218
	FOS-ELM	–	10	–	0.0942	0.1028
Mackey-Glass	MDOS-ELM	Euclidean distance	10	–	0.0827	0.0913
		RHSM	10	0.5	0.0828	0.0973
	OS-ELM	–	–	–	0.1082	0.1242
	FOS-ELM	–	10	–	0.0934	0.1063

Table 15
The RMSE and R^2 of MDOS-ELM models with three common similarity methods on the Magic04 and Twonorn datasets.

Methods of similarity measurement	Magic04		Twonorn	
	Training	Testing	Training	Testing
Euclidean distance	92.72%	90.34%	85.15%	84.94%
Cosine	90.12%	89.84%	88.19%	86.41%
Pearson-Correlation coefficient	89.22%	87.33%	89.91%	89.34%

established by the MDOS-ELM algorithm was better than those of the OS-ELM and the FOS-ELM.

The experiments were conducted using the remaining four regression datasets, and results are shown in Table 14. Table 14 results showed the RMSE or R^2 values obtained by the MDOS-ELM and the MDOS-ELMRHSM for all these cases were always better than those obtained by the OS-ELM and the FOS-ELM, which meant that the generalization capability of the MDOS-ELM was stronger than that of the OS-ELM and the FOS-ELM.

4.2. The experiment on classification datasets

In order to further illustrate the performance of the DMOS-ELM algorithm dealing with the classification problem, fourteen datasets were employed. Among these datasets, Magic04, Adult, Mushroom, Monks-1 and Twonorn are binary classification datasets. Other nine datasets are multi-classification.

4.2.1. Binary classification datasets

Magic04 and Twonorn were analyzed in detail. These two datasets were rather big datasets (Magic04 included 19,020 samples and Twonorn included 7400). The training samples of Magic04 and that of Twonorn were divided into 73 and 21 sample chunks, respectively. Among them, their initial sample chunk both included 1000 samples, and each online updating sample chunks contained 200 samples.

u of the FOS-ELM and that of the MDOS-ELM for Magic04 case respectively was shown in Fig. 3(c) and (d). Fig. 3(c) and (d) displayed that the highest accuracy on the testing set was obtained when $u$were set as 10. The same method is used to set the $u$of other Classification datasets. Both u of the FOS-ELM and that of MDOS-ELM were set as 10 for Twonorn dataset.

Table 15 presented the experimental results of the MDOS-ELM with memory factor calculated by three common similarity methods. The parameter of Huber function was set as different values to experiment when RHSM was referred as to compute similarity. And then results were shown in Table 16. Table 15 indicated that the three common similarities could get better results. For the Magic04 dataset, it was better to use the Euclidean distance method measuring similarity among the three

Table 16
The RMSE and R^2 of MDOS-ELM models with different the RHSM on the Magic04 and Twonorn datasets.

	Magic04		Twonorn	
α	Training	Testing	Training	Testing
0.01	87.72%	87.59%	87.77%	87.72%
0.02	89.12%	88.98%	89.72%	89.72%
0.1	89.22%	89.20%	89.88%	89.89%
0.5	92.17%	92.30%	93.14%	92.97%
1	93.12%	93.05%	92.14%	91.95%
2	89.22%	89.24%	89.20%	89.14%
3	89.62%	89.57%	89.63%	89.41%
5	89.12%	89.03%	88.50%	88.52%

Table 17
Accuracies of the three kinds of model on the Magic04 and Twonorn datasets.

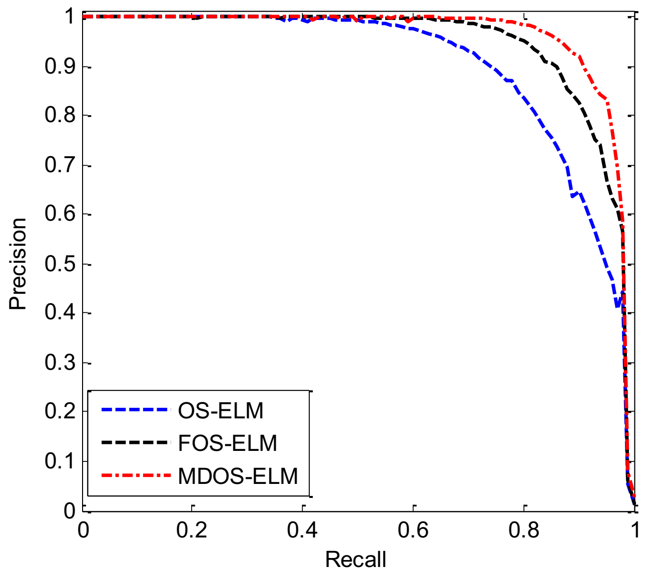
Datasets	Algorithm	Methods of similarity measurement	Accuracy	
			Training	Testing
Magic04	MDOS-ELM	Euclidean distance	92.72%	90.34%
		RHSM	93.12%	93.05%
	OS-ELM	–	82.34%	73.43%
	FOS-ELM	–	88.65%	84.32%
Twonron	MDOS-ELM	Pearson-Correlation coefficient	89.91%	89.34%
		RHSM	93.14%	92.97%
	OS-ELM	–	81.04%	77.34%
	FOS-ELM	–	88.61%	86.21%

Table 18
The confusion matrix for the Magic04 dataset.

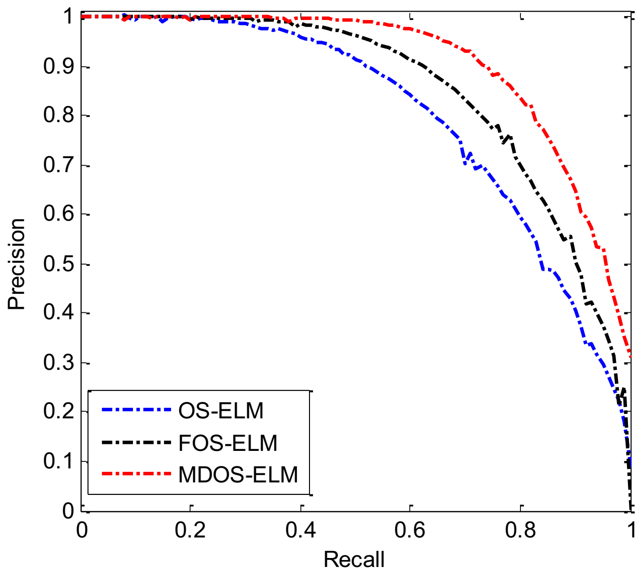
				Actual class	
				Positive	Negative
Predicted class	Positive	OS-ELM	–	10,255	1781
		FOS-ELM	–	10,744	1682
		MDOS-ELM	Euclidean distance	11,422	981
			RHSM	11,502	882
	Negative	OS-ELM	–	2077	4907
		FOS-ELM	–	1588	5006
		MDOS-ELM	Euclidean distance	910	5707
			RHSM	830	5706

Table 19
The confusion matrix for the Twonorn dataset.

				Actual class	
				Positive	Negative
Predicted class	Positive	OS-ELM	–	3324	671
		FOS-ELM	–	3212	519
		MDOS-ELM	Pearson-Correlation coefficient	2984	375
			RHSM	3006	295
	Negative	OS-ELM	–	713	3032
		FOS-ELM	–	485	3184
		MDOS-ELM	Pearson-Correlation coefficient	373	3328
			RHSM	311	3408



(a) Precision-recall for the Magic04



(b) Precision-recall for the Twonorn

Fig. 4. Precision-recall for the Magic04 and Twonorn.

common methods. For Twonorn dataset, Pearson-Correlation coefficient method was better among the three common methods. Table 16 results showed the best performance was obtained when the parameter of Huber function of MDOS-ELM was as $\alpha = 1$ and $\alpha = 0.5$ for the Magic04 dataset and Twonorn datasets, respectively.

Table 17 displayed the comparison results of three kinds of model. Here, the similarity methods were the Euclidean distance and the Pearson-Correlation coefficient for the Magic04 dataset and Twonorn dataset, respectively. The parameters of the Huber function of MDOS-ELM were set as $\alpha = 1$ and $\alpha = 0.5$ for the Magic04 dataset and Twonorn dataset, respectively. Table 17 results showed that the accuracy of the MDOS-ELM was greater than that of the OS-ELM and the FOS-ELM models. The confusion matrices used to measure the performance were shown in Tables 18 and 19.

For the Magic04 dataset, the MDOS-ELM predicted the output of 19,020 samples, among which 17,129 samples achieved the predicted results. The accuracy of the MDOS-ELM was 7.3% higher than that of the FOS-ELM, and 9.6% higher than that of the OS-ELM. For Twonron dataset, the accuracy of the MDOS-ELM was 3.2% higher than that of the FOS-ELM and 7.4% higher than that of The OS-ELM. These curves, plotted the precision recall curves for the three kinds of model, from Fig. 4 also showed that the red curve (MDOS-ELM) was closer to (1,1) point than the other two curves. The experimental results of the three kinds of model for the Adult, Mushroom and Monks-1 datasets were showed in Table 20.

4.2.2. Multi-classification datasets

Nine multi-classification datasets were employed in this section. Breast-Tissue and Glass were analyzed in detail. Both the Breast-Tissue and Glass datasets were a small scale set with only 106 samples and 214, respectively. The training samples of the Breast-Tissue and Glass were divided into 7 and 15 sample chunks respectively. Their initial sample chunks contained 20 samples, and each online updating sample chunk contained 10 samples. Both the u of the MDOS-ELM and that of the FOS-ELM were set as 10 on the Breast-Tissue datasets. And u of the MDOS-ELM and that of the FOS-ELM were set as 2 on the Glass datasets.

Table 21 presented the experimental results of the MDOS-ELM models with memory factor calculated by three common similarity methods. Table 21 results showed that Pearson-Correlation coefficient calculating the similarities was better than others for both Breast-Tissue dataset and Glass dataset. The parameter of Huber function was set as different values to experiment when RHSM was referred as to compute similarity. And then results were shown in Table 22. It demonstrated that the best experimental result was obtained for these two datasets when $\alpha = 0.5$.

Table 23 presented the comparison results of the three kinds of model. Here similarity calculating methods of the MDOS-ELM for these two datasets were the Pearson-Correlation coefficient.

Table 20
Experimental results of the three kinds of model on other Binary classification datasets.

Datasets	Algorithms	Methods of similarity measurement	u	α	Accuracy	
					Training	Testing
Adult	MDOS-ELM	Pearson-Correlation coefficient	10	–	96.51%	94.72%
		RHSM	10	2	96.71%	92.52%
	OS-ELM	–	–	–	93.13%	90.15%
	FOS-ELM	–	10	–	95.12%	91.26%
Mushroom	MDOS-ELM	Person-Correlation-Coefficient	12	–	97.78%	92.31%
		RHSM	12	1	97.41%	93.05%
	OS-ELM	–	–	–	94.12%	90.78%
	FOS-ELM	–	12	–	96.21%	91.22%
Monks-1	MDOS-ELM	Cosine	12	–	96.51%	91.72%
		RHSM	12	0.5	97.04%	93.12%
	OS-ELM	–	–	–	93.11%	90.11%
	FOS-ELM	–	12	–	95.22%	92.17%

Table 21
Accuracy of DDOS-ELM models with three common similarity methods on the Breast-Tissue and Glass datasets.

Methods of similarity measurement	Breast-Tissue		Glass	
	Training	Testing	Training	Testing
Euclidean distance	87.31%	79.18%	79.31%	72.18%
Cosine	88.42%	82.27%	78.43%	72.37%
Pearson-Correlation coefficient	89.21%	87.72%	82.32%	81.26%

Table 22
Accuracy of MDOS-ELM models with the RHSM computing similarities on the Breast-Tissue and Glass datasets.

	Breast-Tissue		Glass	
α	Training	Testing	Training	Testing
0.01	75.13%	74.89%	79.12%	73.15%
0.02	77.41%	76.74%	76.24%	72.37%
0.1	76.25%	75.78%	81.32%	78.23%
0.5	79.45%	79.01%	82.32%	79.14%
1	78.34%	77.61%	79.52%	73.37%
2	75.23%	75.00%	79.02%	72.53%
3	76.32%	75.71%	78.42%	72.34%
5	75.24%	74.63%	77.89%	73.25%

And the parameters of Huber function of the MDOS-ELM were set $\alpha = 0.5$ for these two datasets. For Breast-Tissue dataset, the accuracy of the MDOS-ELM on the training set is 6.57% and 1.88% higher than that of the OS-ELM and the FOS-ELM, respectively. The accuracy of the MDOS-ELM on the testing set is 18.20% and 13.64% higher than that of the OS-ELM and the FOS-ELM, respectively. For Glass dataset, the accuracy of the MDOS-ELM on the training set is 4.6% and 4.43% higher than that of the OS-ELM and the FOS-ELM, respectively. The accuracy of the MDOS-ELM on the testing set is 10.34% and 7.94% higher than that of the OS-ELM and the FOS-ELM, respectively. The experimental results of the three kind of model for other multi-classification were shown in Table 24.

The performance of the MDOS-ELM was tested by using the 23 datasets. The MDOS-ELM with memory factor by different similarly computed methods all could get the expected results. When the RHSM as a similarity calculated the memory factor, the experimental results were better than others. The reason was that Huber function was referenced. Huber function can make a model less sensitive to the outliers. The OS-ELM and the FOS-ELM related online ELM version were employed for comparisons. The results indicated that the accuracy and generalization ability of the model built by the MDOS-ELM was better than those of the OS-ELM and the FOS-ELM.

To summarize, an artificial dataset and twenty-two groups of real-world datasets were used as the experimental data. For the artificial dataset, lower and higher noises were added into the training set and then the experiments were carried out. The experimental results showed that the MDOS-ELM had the better anti-interference ability and robustness than the OS-ELM and the FOS-ELM. The RHSM as a similarity computes the mommy factor, which further improves the robustness of the MDOS-ELM. The reason is that the Huber function is introduced to computer similarity. For regression datasets, the RMSE and the Regression Coefficient R^2 were used to measure the performance of the algorithm. The experimental results showed that the RMSE of the MDOS-ELM were

Table 23
Accuracy of the three kinds of model on the Breast-Tissue and Glass datasets.

Datasets	Algorithm	Methods of similarity measurement	Accuracy	
			Training	Testing
Breast-Tissue	MDOS-ELM	Pearson-Correlation coefficient	89.22%	81.27%
		RHSM	89.43%	82.13%
	OS-ELM	–	82.65%	79.09%
	FOS-ELM	–	87.34%	83.63%
Glass	MDOS-ELM	Pearson-Correlation coefficient	84.25%	81.26%
		RHSM	85.32%	79.13%
	OS-ELM	–	79.65%	72.92%
	FOS-ELM	–	82.32%	73.32%

Table 24
Experimental results of the three kinds of model on other multi-classification datasets.

Datasets	Algorithms	Methods of similarity measurement	u	α	Accuracy	
					Training	Testing
Zoo	MDOS-ELM	Pearson-Correlation coefficient	12	–	86.81%	82.32%
		RHSM	12	1	86.41%	82.34%
	OS-ELM	–	–	–	83.12%	80.17%
	FOS-ELM	–	12	–	85.12%	83.12%
Wine	MDOS-ELM	Person-Correlation-Coefficient	12	–	0.9121	88.32%
		RHSM	12	0.5	92.91%	87.45%
	OS-ELM	–	–	–	89.23%	85.71%
	FOS-ELM	–	12	–	90.31%	87.22%
New-Thyroid	MDOS-ELM	Cosine	10	–	93.46%	90.21%
		RHSM	10	1	93.84%	91.55%
	OS-ELM	–	–	–	90.46%	86.11%
	FOS-ELM	–	10	–	92.86%	89.21%
Image segmentation	MDOS-ELM	Pearson-Correlation coefficient	7	–	93.05%	91.22%
		RHSM	7	1	93.47%	91.65%
	OS-ELM	–	–	–	88.11%	84.91%
	FOS-ELM	–	7	–	89.79%	86.02%
Satellite image	MDOS-ELM	Person-Correlation-Coefficient	5	–	97.81%	93.32%
		RHSM	5	1	97.41%	92.45%
	OS-ELM	–	–	–	94.12%	90.71%
	FOS-ELM	–	5	–	96.12%	91.22%
Vehiche	MDOS-ELM	Cosine	7	–	86.15%	82.37%
		RHSM	7	1	86.64%	82.24%
	OS-ELM	–	–	–	81.03%	79.23%
	FOS-ELM	–	7	–	82.11%	80.22%
Optical digits	MDOS-ELM	Euclidean distance	15	–	95.15%	91.27%
		RHSM	15	1	95.24%	91.72%
	OS-ELM	–	–	–	92.31%	88.15%
	FOS-ELM	–	15	–	94.22%	89.17%

smaller than that of the OS-ELM and the FOS-ELM. And the Regression Coefficient R^2 was larger than that of the OS-ELM and the FOS-ELM. For the classification datasets, the prediction accuracy was used as a measure of the performance of the algorithm. The experimental results showed that the MDOS-ELM had higher prediction accuracy than the OS-ELM and the FOS-ELM.

5. Conclusions

The main contribution of this work is that the proposed MDOS-ELM significantly improves the accuracy and generalization of the FOS-ELM. The MDOS-ELM successfully addresses two problems of online learning. The first one is the timeliness of the training samples and the second is the differences among the valid samples. In the MDOS-ELM, the self-adaptive memory factor is introduced to dynamically adjust the weight between valid samples. The self-adaptive memory factor is determined by two elements. One is the similarity between the new and the old samples, and the other is the prediction error of the current training samples on the previous model. The experimental results demonstrate that the MDOS-ELM has greater accuracy and generalization than those of the FOS-ELM and the OS-ELM on both the classification and the regression datasets. Although the proposed MDOS-ELM with an appropriate period of validity can obtain better performance for online learning, the theoretic analysis to determine the exact period of validity is not very clear. This question will be discussed in future works.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (nos. 61702070, 61425002, 61672121, 61672051, 61572093, 61402066, 61402067, 61370005, 31370778), Program for Changjiang Scholars and Innovative Research Team in University (no. IRT_15R07), the Program for Liaoning Innovative Research Team in University (no. LT2015002), the Basic Research Program of the Key Lab in Liaoning Province Educational Department (nos. LZ2014049, LZ2015004), Scientific Research Fund of Liaoning Provincial Education (nos. L2015015, L2014499), and the Program for Liaoning Key Lab of Intelligent Information Processing and Network Technology in University.

References

[1] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks 4 (2) (1991) 251–257.
[2] G.B. Huang, Learning capability and storage capacity of two-hidden-layer feedforward networks, IEEE Trans. Neural Networks 14 (2) (2003) 274–281.
[3] F. Gruau, D. Whitley, Adding learning to the cellular development of neural networks evolution and the Baldwin effect, Evol. Comput. 1 (3) (1993) 213–233.
[4] T. Chen, H. Chen, Approximation capability to functions of several variables, nonlinear functionals, and operators by radial basis function networks, IEEE Trans. Neural Networks 6 (4) (1995) 904–910.
[5] G.B. Huang, L. Chen, C.K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Networks 17 (4) (2006) 879–892.
[6] G.B. Huang, D.H. Wang, Y. Lan, Extreme learning machines a survey, Int. J. Mach. Learn. Cybern. 2 (2) (2011) 107–122.
[7] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine a new learning scheme of feedforward neural networks, in: 2004 IEEE International Joint Conference on Neural Networks, 2, 2004, pp. 985–990.
[8] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine theory and applications, Neurocomputing 70 (2006) 489–501.
[9] W. Deng, Q. Zheng, L. Chen, Regularized extreme learning machine, in: IEEE Symposium on Computational Intelligence & Data Mining, 2009, pp. 389–395.
[10] X. Lu, L. Ming, W. Liu, H.-X. Li, Probabilistic regularized extreme learning machine for robust modeling of noise data, IEEE Trans. Cybern. PP(99) (2017) 1–10.
[11] X. Liu, L. Wang, G.B. Huang, J. Zhang, J. Yin, Multiple kernel extreme learning machine, Neurocomputing 149 (2015) 253–264.
[12] W. Deng, Q. Zheng, K. Zhang, Reduced kernel extreme learning machine, Adv. Intell. Syst. Comput. 226 (2013) 63–69.
[13] G.B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. 42 (2) (2012) 513–529.
[14] G.B. Huang, M.B. Li, L. Chen, C.K. Siew, Incremental extreme learning machine with fully complex hidden nodes, Neurocomputing 71 (2008) 576–583.
[15] G.B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (2007) 3056–3062.

[16] H. Rong, Y.-S. Ong, A.-H. Tan, Z. Zhu, A fast pruned-extreme learning machine for classification problem, Neurocomputing 72 (2008) 359–366.
[17] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, OP-ELM, optimally pruned extreme learning machine, IEEE Trans. Neural Networks 21 (1) (2010) 158–162.
[18] G. Feng, G.B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Trans. Neural Networks 20 (8) (2009) 1352–1357.
[19] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, American Association for the Advancement of Science, Science 313 (5786) (2006) 504–507.
[20] L.L.C. Kasun, H. Zhou, G.B. Huang, C.-M. Vong, Representational learning with extreme learning machine for big data, IEEE Intell. Syst. 28 (6) (2013) 31–34.
[21] S. Ding, N. Zhang, X. Xu, L. Guo, J. Zhang, Deep extreme learning machine and its application in EEG classification, Math. Probl. Eng. 2015 (2015) 1–11.
[22] C. Wan, Z. Xu, P. Pinson, Z.Y. Dong, K.P. Wong, Probabilistic forecasting of wind power generation using extreme learning machine, IEEE Trans. Power Syst. 29 (3) (2014) 1033–1044.
[23] V. Nikolic, S. Motamedi, S. Shamshirband, D. Petkovic, S. Ch, M. Arif, Extreme learning machine approach for sensorless wind speed estimation, Mechatronics 34 (2015) 78–83.
[24] A. Iosifidis, A. Tefas, I. Pitas, Graph embedded extreme learning machine, IEEE Trans. Cybern. 46 (1) (2016) 331-234.
[25] W. Li, C. Chen, H. Su, Q. Du, Local binary patterns and extreme learning machine for hyperspectral imagery classification, IEEE Trans. Geosci. Remote Sens. 53 (7) (2015) 3681–3693.
[26] W. Zong, G.B. Huang, Face recognition based on extreme learning machine, Neurocomputing 74 (16) (2011) 2541–2551.
[27] S. Shamshirband, K. Mohammadi, H.-L. Chen, G.N. Samy, D. Petković, C. Ma, Daily global solar radiation prediction from air temperatures using kernel extreme learning machine, a case study for Iran, J. Atmos. Solar-Terr. Phys. 134 (2015) 109–117.
[28] R. Zhang, G.B. Huang, N. Sundararajan, P. Saratchandran, Multi-category classification using an extreme learning machine for microarray gene expression cancer diagnosis, IEEE/ACM Trans. Comput. Biol. Bioinf. 4 (3) (2007) 485–495.
[29] S.-H. Wang, T.-M. Zhan, Y. Chen, Y. Zhang, M. Yang, Multiple sclerosis detection based on biorthogonal wavelet transform, rbf kernel principal component analysis, and logistic regression, IEEE Access 4 (2016) 7567–7576.
[30] S. Lu, Z. Lu, J. Yang, M. Yang, S. Wang, A pathological brain detection system based on kernel based ELM, Multimed. Tools Appl. (2016) 1–14, doi:10.1007/s11042-016-3559-z.
[31] S. Lu, X. Qiu, J. Shi, N. Li, Z.-H. Lu, P. Chen, et al., A pathological brain detection system based on extreme learning machine optimized by bat algorithm, CNS Neurol. Disord. – Drug Targets 16 (1) (2016) 23–29.
[32] Y.-D. Zhang, G. Zhao, J. Sun, X. Wu, Z.-H. Wang, H.-M. Liu, V.V. Govindaraj, et al., Smart pathological brain detection by synthetic minority oversampling technique, extreme learning machine, and jaya algorithm, Multimed. Tools Appl. (2017) 1–20, doi:10.1007/s11042-017-5023-0.
[33] A. Sorjamaa, Y. Miche, R. Weiss, A. Lendasse, Long-term prediction of time series using NNE-based projection and OP-ELM, in: IEEE International Joint Conference on Neural Networks, 2008, pp. 2674–2680.
[34] N.A. Shrivastava, B.K. Panigrahi, A hybrid wavelet-ELM based short term price forecasting for electricity markets, Int. J. Electr. Power Energy Syst. 55 (7) (2014) 41–50.
[35] A.R. Lima, A.J. Cannon, W.W. Hsieh, Forecasting daily streamflower using online sequential extreme learning machines, J. Hydrol. 537 (2016) 431–443.
[36] F. Silva, P. Urbano, L. Correia, A.L. Christensen, odNEAT, an algorithm for decentralised online evolution of robotic controllers, Evol. Comput. 23 (3) (2012) 421–449.
[37] N.Y. Liang, G.B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Trans. Neural Networks 17 (6) (2006) 1411–1423.
[38] H.J. Rong, G.B. Huang, N. Sundararajan, P. Saratchandran, Online sequential fuzzy extreme learning machine for function approximation and classification problems, IEEE Trans. Syst. Man Cybern. Part B Cybern. Publ. IEEE Syst. Man Cybern. Soc. 39 (4) (2009) 1067–1070.
[39] S. Scardapane, D. Comminiello, M. Scarpiniti, A. Uncini, Online sequential extreme learning machine with kernels, IEEE Trans. Neural Networks Learn. Syst. 26 (9) (2015) 2214–2220.
[40] Y. Lan, Y.C. Soh, G.B. Huang, Letters ensemble of online sequential extreme learning machine, Neurocomputing 72 (2009) 3391–3395.
[41] J. Zhao, Z. Wang, S.P. Dong, Online sequential extreme learning machine with forgetting mechanism, Neurocomputing 87 (15) (2012) 79–89.
[42] F.A.A. Souza, R.A.J. Mendes, Review of soft sensor methods for regression applications, Chemom. Intell. Lab. Syst. 152 (15) (2016) 69–79.
[43] K. Kishia, E. Kitagawab, N. Onikuraa, A. Nakamuraa, H. Iwahashib, Expression analysis of sex-specific and 17beta-estradiol-responsive genes in the Japanese medaka, Oryzias latipes, using oligonucleotide microarrays, Genomics 88 (2) (2006) 241–251.
[44] A. Ghaffari, E. Fatemizadeh, Robust Huber similarity measure for image registration in the presence of spatially-varying intensity distortion, Signal Process. 109 (2015) 54–68.
[45] J Huber Peter, Robust estimation of a location parameter, Ann. Math. Stat. 35 (1) (1964) 73–101.
[46] C.Y. Deng, A generalization of the Sherman–Morrison–Woodbury formula, Appl. Math. Lett. 24 (9) (2011) 1561–1564.
[47] T. Suzuki, M. Sugiyama, Sufficient dimension reduction via squared-loss mutual information estimation, Neural Comput. 25 (3) (2010) 725–758.
[48] C.-H. Xie, G.-H. Yang, Cooperative guaranteed cost fault-tolerant control for multi-agent systems with time-varying actuator faults, Neurocomputing 214 (2016) 382-290.
[49] H. Xu, S. Mannor, Robustness and generalization, Mach. Learn. 86 (3) (2012) 391–423.

Quan-Yi Zou is currently working towards the M.Sc. degree in computer science & technology from Dalian University, Dalian, People's Republic of China. His research interests include machine learning and neural networks.

Xiao-Jun Wang received the B.S. degree in automation from Dalian Ocean University, Dalian, China, in 2009. She received the M.Sc. and Ph.D. degrees in control theory & control engineering from Northeastern University, Shenyang, China, in 2011 and 2016, respectively. Her current research interests include machine learning, modeling, and artificial intelligence.

Chang-Jun Zhou received the Ph.D. degree in Dalian University of Technology, Dalian, in 2008. Currently, he is a Professor at Dalian University, Dalian, People's Republic of China. He was a recipient of Liaoning distinguished professor in 2014, and his research interests include intelligence computing and pattern recognition.

Qiang Zhang received the Ph.D. degree in Xidian University, Xian, in 2002. Currently, he is a Professor at Dalian University, Dalian. He is the National Science Fund for Distinguished Young Scholars, and his research interests include intelligence computing, neural networks, DNA Computing and Computer Animation.