Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 938548, 11 pages
http://dx.doi.org/10.1155/2014/938548

Research Article

Online Regularized and Kernelized Extreme Learning Machines with Forgetting Mechanism

Xinran Zhou, Zijian Liu, and Congxu Zhu

School of Information Science and Engineering, Central South University, Changsha 410083, China

Correspondence should be addressed to Zijian Liu; liuzijian2015@126.com

Received 8 April 2014; Revised 25 May 2014; Accepted 17 June 2014; Published 10 July 2014

Academic Editor: Ching-Ter Chang

To apply the single hidden-layer feedforward neural networks (SLFN) to identify time-varying system, online regularized extreme learning machine (ELM) with forgetting mechanism (FORELM) and online kernelized ELM with forgetting mechanism (FOKELM) are presented in this paper. The FORELM updates the output weights of SLFN recursively by using Sherman-Morrison formula, and it combines advantages of online sequential ELM with forgetting mechanism (FOS-ELM) and regularized online sequential ELM (ReOS-ELM); that is, it can capture the latest properties of identified system by studying a certain number of the newest samples and also can avoid issue of ill-conditioned matrix inversion by regularization. The FOKELM tackles the problem of matrix expansion of kernel based incremental ELM (KB-IELM) by deleting the oldest sample according to the block matrix inverse formula when samples occur continually. The experimental results show that the proposed FORELM and FOKELM have better stability than FOS-ELM and have higher accuracy than ReOS-ELM in nonstationary environments; moreover, FORELM and FOKELM have time efficiencies superiority over dynamic regression extreme learning machine (DR-ELM) under certain conditions.

1. Introduction

Plenty of research work has shown that the single hidden-layer feedforward neural networks (SLFN) can approximate any function and form decision boundaries with arbitrary shapes if the activation function is chosen properly [1–3]. However, most of traditional approaches (such as BP algorithm) for training SLFN are slow due to their iterative steps. To train SLFN fast, Huang et al. proposed a learning algorithm called extreme learning machine (ELM), which randomly assigns the hidden nodes parameters (the input weights and hidden layer biases of additive networks or the centers and impact factors of RBF networks) and then determines the output weights by the Moore-Penrose generalized inverse [4, 5]. The original ELM is batch learning algorithm.

For some practical fields where the training data are generated gradually, online sequential learning algorithms are preferred over batch learning algorithms as sequential learning algorithms do not require retraining whenever a new sample is received. Hence, Liang et al. developed a kind of online sequential ELM (OS-ELM) using the recursive least square [6]. OS-ELM for SLFN produced better generalization performance at faster learning speed compared with the previous sequential learning algorithms. Moreover, for time-varying environments, recently several incremental sequential ELMs are presented; they apply constant or adaptive forgetting factor [7, 8] or iteration approach [9] to strengthen new sample's contribution on model. Speaking theoretically, they cannot eliminate old samples' effect on model thoroughly. To let ELM study the latest properties of identified object, Zhao et al. developed online sequential ELM with forgetting mechanism (FOS-ELM) [10]. Fixed-memory extreme learning machine (FM-ELM) of Zhang and Wang [11] can be thought of as a special case of FOS-ELM with the parameter p in [10] being 1. Although experimental results show FOS-ELM has higher accuracy [10], it may encounter the matrix singularity problem and run unstably.

As a variant of ELM, regularized ELM (RELM) [12–14], which is equivalent to the constrained optimization based ELM [15, 16] mathematically, absorbing the thought of structural risk minimization of statistical learning theory [17], can overcome the overfitting problem of ELM and provides better generalization ability than original ELM when noises or outliers exist in the dataset [12]. Furthermore, the regularized OS-ELM (ReOS-ELM) developed by Huynh and Won [13], which is equivalent to sequential regularized ELM (SRELM) [14] and least square incremental ELM (LS-IELM) [18] essentially, can avoid singularity problem.

If the feature mapping in SLFN is unknown to users, the kernel based ELM (KELM) can be constructed [15, 16]. For the application where samples arrive gradually, Guo et al. developed kernel based incremental ELM (KB-IELM) [18].

However, in time-varying or nonstationary applications, the newer training data usually carry more information about systems, and the older ones possibly carry less information, even misleading information; that is, the training samples usually have timeliness. The ReOS-ELM and KB-IELM cannot reflect the timeliness of sequential training data well. On the other hand, if a huge number of samples emerge, for KB-IELM, the required storage space for matrix will infinitely increase with learning ongoing and new samples arriving ceaselessly, and at last storage overflow will happen necessarily, so KB-IELM cannot be utilized at all under the circumstances.

In this paper, we combine advantages of FOS-ELM and ReOS-ELM and propose online regularized ELM with forgetting mechanism (FORELM) for time-varying applications. FORELM can overcome the potential matrix singularity problem by using regularization and eliminate effections of the outdated data on model by incorporating forgetting mechanism. Like FOS-ELM, the ensemble skill also may be employed in FORELM to enhance its stability; that is, FORELM comprises p ($p \geq 1$) ReOE-ELMs with forgetting mechanism, each of which trains a SLFN; the average of those outputs represents the final output of the ensemble of these SLFN. Additionally, forgetting mechanism also is incorporated into KB-IELM and online kernelized ELM with forgetting mechanism (FOKELM) is presented, which can deal with the problem of matrix expansion of KB-IELM. The designed FORELM and FOKELM update model recursively. The experimental results show the better performance of FORELM and FOKELM approach in nonstationary environments.

It should be noted that our methods adjust the output weights of SLFN due to addition and deletion of the samples one by one, namely, learn and forget samples sequentially, and network architecture is fixed. They are completely different from those offline incremental ELMs (I-ELM) [19–21] and incremental RELM [22] seeking optimal network architecture by adding hidden nodes one by one and learning the data in batch mode.

The rest of this paper is organized as follows. Section 2 gives a brief review of the basic concepts and related works of ReOS-ELM and KB-IELM. Section 3 proposes new online learning algorithms, namely, FORELM and FOKELM. Performance evaluation is conducted in Section 4. Conclusions are drawn in Section 5.

2. Brief Review of the ReOS-ELM and KB-IELM

For simplicity, ELM based learning algorithm for SLFN with multiple input single output is discussed.

The output of a SLFN with L hidden nodes (additive or RBF nodes) can be represented by

$$f(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta}, \quad \mathbf{x} \in R^n, \ \mathbf{a}_i \in R^n, \quad (1)$$

where $\mathbf{a}_i$ and b_i are the learning parameters of hidden nodes, $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_L]^{\mathrm{T}}$ is the vector of the output weights, and $G(\mathbf{a}_i, b_i, \mathbf{x})$ denotes the output of the ith hidden node with respect to the input $\mathbf{x}$, that is, activation function. $\mathbf{h}(\mathbf{x}) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \ldots, G(\mathbf{a}_L, b_L, \mathbf{x})]$ is a feature mapping from the n-dimensional input space to the L-dimensional hidden-layer feature space. In ELM, $\mathbf{a}_i$ and b_i are randomly determined firstly.

For a given set of distinct training data $\{(\mathbf{x}_i, y_i)\}_1^N \subset R^n \times R$, where $\mathbf{x}_i$ is an n-dimensional input vector and y_i is the corresponding scalar observation, the RELM, that is, constrained optimization based ELM, can be formulated as

$$\text{Minimize:} \quad L_{P_{\mathrm{ELM}}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + c\frac{1}{2}\|\boldsymbol{\xi}\|^2, \quad (2a)$$

$$\text{Subject to:} \quad \mathbf{H}\boldsymbol{\beta} = \mathbf{Y} - \boldsymbol{\xi}, \quad (2b)$$

where $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_N]^{\mathrm{T}}$ denotes the training error. $\mathbf{Y} = [y_1, y_2, \ldots, y_N]^{\mathrm{T}}$ indicates the target value of all the samples. $\mathbf{H}_{N\times L} = [\mathbf{h}(\mathbf{x}_1)^{\mathrm{T}}, \ldots, \mathbf{h}(\mathbf{x}_N)^{\mathrm{T}}]^{\mathrm{T}}$ is the mapping matrix for the inputs of all the samples. c is the regularization parameter (a positive constant).

Based on the KKT theorem, the constrained optimization of (2a) and (2b) can be transferred to the following dual optimization problem:

$$L_{D_{\mathrm{ELM}}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + c\frac{1}{2}\|\boldsymbol{\xi}\|^2 - \boldsymbol{\alpha}(\mathbf{H}\boldsymbol{\beta} - \mathbf{Y} + \boldsymbol{\xi}), \quad (3)$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]$ is the Lagrange multipliers vector. Using KKT optimality conditions, the following equations can be obtained:

$$\frac{\partial L_{D_{\mathrm{ELM}}}}{\partial \boldsymbol{\beta}} = \mathbf{0} \longrightarrow \boldsymbol{\beta} = \mathbf{H}^{\mathrm{T}}\mathbf{a}, \quad (4a)$$

$$\frac{\partial L_{D_{\mathrm{ELM}}}}{\partial \boldsymbol{\xi}} = \mathbf{0} \longrightarrow \mathbf{a} = c\boldsymbol{\xi}, \quad (4b)$$

$$\frac{\partial L_{D_{\mathrm{ELM}}}}{\partial \boldsymbol{\alpha}} = \mathbf{0} \longrightarrow \mathbf{H}\boldsymbol{\beta} - Y + \boldsymbol{\xi} = \mathbf{0}. \quad (4c)$$

Ultimately, $\boldsymbol{\beta}$ can be obtained as follows [12, 15, 16]:

$$\boldsymbol{\beta} = \left(c^{-1}\mathbf{I} + \mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{Y} \quad (5a)$$

$$\text{or} \quad \boldsymbol{\beta} = \mathbf{H}^{\mathrm{T}}\left(c^{-1}\mathbf{I} + \mathbf{H}\mathbf{H}^{\mathrm{T}}\right)^{-1}\mathbf{Y}. \quad (5b)$$

In order to reduce computational costs, when $N > L$, one may prefer to apply solution (5a), and when $N < L$, one may prefer to apply solution (5b).

If the feature mapping $\mathbf{h}(\mathbf{x})$ is unknown, one can apply Mercer's conditions on RELM. The kernel matrix is defined as $\mathbf{\Omega} = \mathbf{H}\mathbf{H}^{\mathrm{T}}$ and $\Omega_{ij} = \mathbf{h}(\mathbf{x}_i)\mathbf{h}(\mathbf{x}_j)^{\mathrm{T}} = K(\mathbf{x}_i, \mathbf{x}_j)$. Then, the output of SLFN by kernel based RELM can be given as

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\mathbf{H}^{\mathrm{T}}\left(c^{-1}\mathbf{I} + \mathbf{H}\mathbf{H}^{\mathrm{T}}\right)^{-1}\mathbf{Y} \\ &= \left[K(\mathbf{x}, \mathbf{x}_1), K(\mathbf{x}, \mathbf{x}_2), \ldots, K(\mathbf{x}, \mathbf{x}_N)\right]\left(c^{-1}\mathbf{I} + \mathbf{\Omega}\right)^{-1}\mathbf{Y}. \end{aligned} \tag{6}$$

2.1. ReOS-ELM. The ReOS-ELM, that is, SRELM and LS-IELM, can be retold as follows.

For time $k-1$, let $\mathbf{h}_i = \mathbf{h}(\mathbf{x}_i)$, $\mathbf{H}_{k-1} = \left[\mathbf{h}_s^{\mathrm{T}}\ \mathbf{h}_{s+1}^{\mathrm{T}}\ \cdots\ \mathbf{h}_{k-1}^{\mathrm{T}}\right]^{\mathrm{T}}$, $\mathbf{Y}_{k-1} = \left[y_s\ y_{s+1}\ \cdots\ y_{k-1}\right]^{\mathrm{T}}$; then, according to (5a), solution of RELM can be expressed as

$$\mathbf{P}_{k-1} = \left(c^{-1}\mathbf{I} + \mathbf{H}_{k-1}^{\mathrm{T}}\mathbf{H}_{k-1}\right)^{-1}, \tag{7a}$$

$$\boldsymbol{\beta}_{k-1} = \mathbf{P}_{k-1}\mathbf{H}_{k-1}^{\mathrm{T}}\mathbf{Y}_{k-1}. \tag{7b}$$

For time k, the new sample $(\mathbf{x}_k, y_k)$ arrives; thus $\mathbf{H}_k = \left[\mathbf{H}_{k-1}^{\mathrm{T}}\ \mathbf{h}_k^{\mathrm{T}}\right]^{\mathrm{T}}, \mathbf{Y}_k = \left[\mathbf{Y}_{k-1}^{\mathrm{T}}\ y_k\right]^{\mathrm{T}}$; applying Sherman-Morrison-Woodbury (SMW) formula [23], the current $\mathbf{P}$ and $\boldsymbol{\beta}$ can be computed as

$$\begin{aligned} \mathbf{P}_k &= \left(c^{-1}\mathbf{I} + \mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k\right)^{-1} = \left(c^{-1}\mathbf{I} + \mathbf{H}_{k-1}^{\mathrm{T}}\mathbf{H}_{k-1} + \mathbf{h}_k^{\mathrm{T}}\mathbf{h}_k\right)^{-1} \\ &= \left(\mathbf{P}_{k-1}^{-1} + \mathbf{h}_k^{\mathrm{T}}\mathbf{h}_k\right)^{-1} = \mathbf{P}_{k-1} - \frac{\mathbf{P}_{k-1}\mathbf{h}_k^{\mathrm{T}}\mathbf{h}_k\mathbf{P}_{k-1}}{1 + \mathbf{h}_k\mathbf{P}_{k-1}\mathbf{h}_k^{\mathrm{T}}}, \end{aligned} \tag{8a}$$

$$\begin{aligned} \boldsymbol{\beta}_k &= \mathbf{P}_k\mathbf{H}_k^{\mathrm{T}}\mathbf{Y}_k = \mathbf{P}_k\left(\mathbf{H}_{k-1}^{\mathrm{T}}\mathbf{Y}_{k-1} + \mathbf{h}_k^{\mathrm{T}}y_k\right) \\ &= \mathbf{P}_k\left(\mathbf{P}_{k-1}^{-1}\mathbf{P}_{k-1}\mathbf{H}_{k-1}^{\mathrm{T}}\mathbf{Y}_{k-1} + \mathbf{h}_k^{\mathrm{T}}y_k\right) \\ &= \mathbf{P}_k\left(\left(\mathbf{P}_k^{-1} - \mathbf{h}_k^{\mathrm{T}}\mathbf{h}_k\right)\boldsymbol{\beta}_{k-1} + \mathbf{h}_k^{\mathrm{T}}Y_k\right) \\ &= \boldsymbol{\beta}_{k-1} + \mathbf{P}_k\mathbf{h}_k^{\mathrm{T}}\left(y_k - \mathbf{h}_k\boldsymbol{\beta}_{k-1}\right). \end{aligned} \tag{8b}$$

2.2. KB-IELM. For time $k-1$, let

$$\begin{aligned} &\mathbf{A}_{k-1} \\ &= \left(c^{-1}\mathbf{I} + \mathbf{\Omega}_{k-1}\right) \\ &= \begin{bmatrix} c^{-1} + K(\mathbf{x}_s, \mathbf{x}_s) & K(\mathbf{x}_s, \mathbf{x}_{s+1}) & \cdots & K(\mathbf{x}_s, \mathbf{x}_{k-1}) \\ K(\mathbf{x}_{s+1}, \mathbf{x}_s) & c^{-1} + K(\mathbf{x}_{s+1}, \mathbf{x}_{s+1}) & \cdots & K(\mathbf{x}_{s+1}, \mathbf{x}_{k-1}) \\ \vdots & \vdots & \vdots & \vdots \\ K(\mathbf{x}_{k-1}, \mathbf{x}_s) & K(\mathbf{x}_{k-1}, \mathbf{x}_{s+1}) & \cdots & c^{-1} + K(\mathbf{x}_{k-1}, \mathbf{x}_{k-1}) \end{bmatrix}. \end{aligned} \tag{9}$$

For time k, the new sample $(\mathbf{x}_k, y_k)$ arrives; thus

$$\mathbf{A}_k = c^{-1}\mathbf{I} + \mathbf{\Omega}_k = \begin{bmatrix} \mathbf{A}_{k-1} & \mathbf{V}_k \\ \mathbf{V}_k^{\mathrm{T}} & v_k \end{bmatrix}, \tag{10}$$

where $\mathbf{V}_k = \left[K(\mathbf{x}_k, \mathbf{x}_s), K(\mathbf{x}_k, \mathbf{x}_{s+1}), \ldots, K(\mathbf{x}_k, \mathbf{x}_{k-1})\right]^{\mathrm{T}}$, $v_k = K(\mathbf{x}_k, \mathbf{x}_k) + 1/c$. Using the block matrix inverse formula [23], $\mathbf{A}_k^{-1}$ can be calculated from $\mathbf{A}_{k-1}^{-1}$ as

$$\begin{aligned} \mathbf{A}_k^{-1} &= \begin{bmatrix} \mathbf{A}_{k-1} & \mathbf{V}_k \\ \mathbf{V}_k^{\mathrm{T}} & v_k \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \mathbf{A}_{k-1}^{-1} + \mathbf{A}_{k-1}^{-1}\mathbf{V}_k\rho_k^{-1}\mathbf{V}_{k+1}^{\mathrm{T}}\mathbf{A}_{k-1}^{-1} & -\mathbf{A}_{k-1}^{-1}\mathbf{V}_t\rho_k^{-1} \\ -\rho_k^{-1}\mathbf{V}_k^{\mathrm{T}}\mathbf{A}_{k-1}^{-1} & \rho_k^{-1} \end{bmatrix}, \end{aligned} \tag{11}$$

where $\rho_k = v_k - \mathbf{V}_k^{\mathrm{T}}\mathbf{A}_{k-1}^{-1}\mathbf{V}_k$.

3. The Proposed FORELM and FOKELM

When SLFN is employed to model online for time-varying system, training samples are not only generated one by one, but also often have the property of timeliness; that is, training data have a period of validity. Therefore, during the learning process by online sequential learning algorithm, the older or outdated training data, whose effectiveness is less or is lost after several unit times, should be abandoned, which is the idea of forgetting mechanism [10]. ReOS-ELM (i.e., SRELM or LS-IELM) and KB-IELM cannot reflect the timeliness of sequential training data. In this section, the forgetting mechanism is added to them to eliminate the outdated data that might have misleading or bad effect on built SLFN. On the other hand, for KB-IELM, to abandon samples can prevent matrix $\mathbf{A}^{-1}$ of (11) from expanding infinitely. The computing procedures of deleting sample are given, and the completed online regularized ELM and kernelized ELM with forgetting mechanism are presented.

3.1. Decremental RELM and FORELM. After RELM has studied a given number z of samples and SLFN has been applied for prediction, RELM would discard the oldest sample $(\mathbf{x}_s, y_s)$ from samples set.

Let $\overline{\mathbf{H}} = \left[\mathbf{h}_{s+1}^{\mathrm{T}}\ \mathbf{h}_{s+2}^{\mathrm{T}}\ \cdots\ \mathbf{h}_k^{\mathrm{T}}\right]^{\mathrm{T}}$, $\overline{\mathbf{Y}} = \left[y_{s+1}\ y_{s+2}\ \cdots\ y_k\right]^{\mathrm{T}}$; then $\mathbf{H}_k = \left[\mathbf{h}_s^{\mathrm{T}}\ \overline{\mathbf{H}}^{\mathrm{T}}\right]^{\mathrm{T}}$, $\mathbf{Y}_k = \left[y_s\ \overline{\mathbf{Y}}^{\mathrm{T}}\right]^{\mathrm{T}}$, and

$$\begin{aligned} \overline{\mathbf{P}} &= \left(c^{-1}\mathbf{I} + \overline{\mathbf{H}}^{\mathrm{T}}\overline{\mathbf{H}}\right)^{-1} = \left(c^{-1}\mathbf{I} + \mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k - \mathbf{h}_s^{\mathrm{T}}\mathbf{h}_s\right)^{-1} \\ &= \left(\mathbf{P}_k^{-1} - \mathbf{h}_s^{\mathrm{T}}\mathbf{h}_s\right)^{-1}. \end{aligned} \tag{12}$$

Furthermore, using SMW formula, then

$$\overline{\mathbf{P}} = \mathbf{P}_k + \frac{\mathbf{P}_k\mathbf{h}_s^{\mathrm{T}}\mathbf{h}_s\mathbf{P}_k}{1 - \mathbf{h}_s\mathbf{P}_k\mathbf{h}_s^{\mathrm{T}}}. \tag{13}$$

Moreover,

$$\begin{aligned} \overline{\boldsymbol{\beta}} &= \overline{\mathbf{P}}\,\overline{\mathbf{H}}^{\mathrm{T}}\overline{\mathbf{Y}} = \overline{\mathbf{P}}\left(\mathbf{H}_k^{\mathrm{T}}\mathbf{Y}_k - \mathbf{h}_s^{\mathrm{T}}y_s\right) = \overline{\mathbf{P}}\left(\mathbf{P}_k^{-1}\mathbf{P}_k\mathbf{H}_k^{\mathrm{T}}\mathbf{Y}_k - \mathbf{h}_s^{\mathrm{T}}y_s\right) \\ &= \overline{\mathbf{P}}\left(\left(\overline{\mathbf{P}}^{-1} + \mathbf{h}_s^{\mathrm{T}}\mathbf{h}_s\right)\boldsymbol{\beta}_k - \mathbf{h}_s^{\mathrm{T}}y_s\right) = \boldsymbol{\beta}_k + \overline{\mathbf{P}}\mathbf{h}_s^{\mathrm{T}}\left(\mathbf{h}_s\boldsymbol{\beta}_k - y_s\right). \end{aligned} \tag{14}$$

Next time, $\mathbf{P}_{k+1}$ and $\boldsymbol{\beta}_{k+1}$ can be calculated from $\overline{\mathbf{P}}$ (viewed as $\mathbf{P}_k$) and $\overline{\boldsymbol{\beta}}$ (viewed as $\boldsymbol{\beta}_k$) according to (8a) and (8b), respectively.

Suppose that FORELM may consist of p ReOS-ELMs with forgetting mechanism, by which SFLNs trained have the same output of hidden node $G(\mathbf{a}, b, \mathbf{x})$ and the same number of hidden nodes L. In the following FORELM algorithm, the variables and parameters with superscript r are relevant to the rth SFLN to be trained by the rth ReOS-ELM with forgetting mechanism. Synthesize ReOS-ELM and the decremental RELM; then we can get FORELM as follows.

Step 1. Initialization:

(1) Choose the hidden output function $G(\mathbf{a}, b, \mathbf{x})$ of SFLN with the certain activation function and the number of hidden nodes L. Set the value of p.

(2) Randomly assign hidden parameters $(\mathbf{a}_i^r, b_i^r)$, $i = 1, 2, \ldots, L$, $r = 1, 2, \ldots, p$.

(3) Determine c; set $\mathbf{P}_{k-1}^r = c\mathbf{I}_{L\times L}$.

Step 2. Incrementally learn initial $z-1$ samples; that is, repeat the following procedure for $z-1$ times:

(1) Get current sample $(\mathbf{x}_k, y_k)$.

(2) Calculate $\mathbf{h}_k^r$ and $\mathbf{P}_k^r$: $\mathbf{h}_k^r = [G(a_1^r, b_1^r, x_k), \ldots, G(a_L^r, b_L^r, x_k)]$; calculate $\mathbf{P}_k^r$ by (8a).

(3) Calculate $\boldsymbol{\beta}_k^r$: if sample $(\mathbf{x}_k, y_k)$ is the first one, then $\boldsymbol{\beta}_k^r = \mathbf{P}_k^r(\mathbf{h}_k^r)^{\mathrm{T}} y_k$, else calculate $\boldsymbol{\beta}_k^r$ by (8b).

Step 3. Online modeling and prediction: repeat the following procedure during every step:

(1) Acquire current y_k, form new sample $(\mathbf{x}_k, y_k)$, and calculate $\mathbf{h}_k^r$, $\mathbf{P}_k^r$, and $\boldsymbol{\beta}_k^r$ by (8a) and (8b).

(2) Prediction: form $\mathbf{x}_{k+1}$, the output of the rth SFLN; that is, prediction $\hat{y}_{k+1}^r$ of y_{k+1} can be calculated by (1): $\hat{y}_{k+1}^r = \mathbf{h}^r(\mathbf{x}_{k+1})\boldsymbol{\beta}_k^r$, the final prediction $\hat{y}_{k+1} = \sum_{r=1}^{p} \hat{y}_{k+1}^r / p$.

(3) Delete the oldest sample $(\mathbf{x}_s, y_s)$: calculate $\overline{\mathbf{P}}^r$ and $\overline{\boldsymbol{\beta}}^r$ by (13) and (14), respectively.

3.2. Decremental KELM and FOKELM. After KELM has studied the given number z of samples and SLFN has been applied for prediction, KELM would discard the oldest sample $(\mathbf{x}_s, y_s)$ from samples set.

Let $\overline{\boldsymbol{\Omega}}_{ij} = K(\mathbf{x}_{s+i}, \mathbf{x}_{s+j})$, $i, j = 1, 2, \ldots, k-s$. $\overline{\mathbf{A}} = \overline{\boldsymbol{\Omega}} + c^{-1}\mathbf{I}$, $\overline{\mathbf{V}} = [K(\mathbf{x}_s, \mathbf{x}_{s+1}), \ldots, K(\mathbf{x}_s, \mathbf{x}_k)]$, $\overline{v} = K(\mathbf{x}_s, \mathbf{x}_s) + 1/c$; then $\mathbf{A}_k$ can be written in the following partitioned matrix form:

$$\mathbf{A}_k = \boldsymbol{\Omega} + c^{-1}\mathbf{I} = \begin{bmatrix} \overline{v} & \overline{\mathbf{V}} \\ \overline{\mathbf{V}}^{\mathrm{T}} & \overline{\mathbf{A}} \end{bmatrix}. \tag{15}$$

Moreover, using the block matrix inverse formula, such equation can be obtained:

$$\begin{aligned} \mathbf{A}_k^{-1} &= \boldsymbol{\Omega} + c^{-1}\mathbf{I} = \begin{bmatrix} \overline{v} & \overline{\mathbf{V}} \\ \overline{\mathbf{V}}^{\mathrm{T}} & \overline{\mathbf{A}} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \overline{\rho}^{-1} & -\overline{\rho}^{-1}\overline{\mathbf{V}}\,\overline{\mathbf{A}}^{-1} \\ -\overline{\mathbf{A}}^{-1}\overline{\mathbf{V}}^{\mathrm{T}}\overline{\rho}^{-1} & \overline{\mathbf{A}}^{-1} + \overline{\mathbf{A}}^{-1}\overline{\mathbf{V}}^{\mathrm{T}}\overline{\rho}^{-1}\overline{\mathbf{V}}\,\overline{\mathbf{A}}^{-1} \end{bmatrix}, \end{aligned} \tag{16}$$

where $\overline{\rho} = \overline{v} - \overline{\mathbf{V}}\,\overline{\mathbf{A}}^{-1}\overline{\mathbf{V}}^{\mathrm{T}}$.

Rewrite $\mathbf{A}_k^{-1}$ in the partitioned matrix form as

$$\mathbf{A}_k^{-1} = \begin{bmatrix} \mathbf{A}_{k\ (1,1)}^{-1} & \mathbf{A}_{k\ (1,2:\mathrm{end})}^{-1} \\ \mathbf{A}_{k\ (2:\mathrm{end},1)}^{-1} & \mathbf{A}_{k\ (2:\mathrm{end},2:\mathrm{end})}^{-1} \end{bmatrix}. \tag{17}$$

Compare (16) and (17); $\overline{\mathbf{A}}^{-1}$ can be calculated as

$$\begin{aligned} \overline{\mathbf{A}}^{-1} &= \mathbf{A}_{k\ (2:\mathrm{end},2:\mathrm{end})}^{-1} - \overline{\mathbf{A}}^{-1}\overline{\mathbf{V}}^{\mathrm{T}}\overline{\rho}^{-1}\overline{\mathbf{V}}\,\overline{\mathbf{A}}^{-1} \\ &= \mathbf{A}_{k\ (2:\mathrm{end},2:\mathrm{end})}^{-1} - \frac{\left(-\overline{\mathbf{A}}^{-1}\overline{\mathbf{V}}^{\mathrm{T}}\overline{\rho}^{-1}\right)\left(-\overline{\rho}^{-1}\overline{\mathbf{V}}\,\overline{\mathbf{A}}^{-1}\right)}{\overline{\rho}^{-1}} \\ &= \mathbf{A}_{k\ (2:\mathrm{end},2:\mathrm{end})}^{-1} - \frac{\mathbf{A}_{k\ (2:\mathrm{end},1)}^{-1} \times \mathbf{A}_{k\ (1,2:\mathrm{end})}^{-1}}{\mathbf{A}_{k\ (1,1)}^{-1}}. \end{aligned} \tag{18}$$

Next time, compute $\mathbf{A}_{k+1}^{-1}$ from $\overline{\mathbf{A}}^{-1}$ (viewed as $\mathbf{A}_k^{-1}$) according to (11).

Integrate KB-IELM with the decremental KELM; further, we can obtain FOKELM as follows.

Step 1. Initialization: choose kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ with corresponding parameter values, and determine c.

Step 2. Incrementally learn initial $z-1$ samples: calculate $\mathbf{A}_k^{-1}$: if there exists a sample only, then $\mathbf{A}_k^{-1} = 1/(K(\mathbf{x}_k, \mathbf{x}_k) + c^{-1})$, else calculate $\mathbf{A}_k^{-1}$ by (11).

Step 3. Online modeling and prediction:

(1) Acquire new sample $(\mathbf{x}_k, y_k)$ and calculate $\mathbf{A}_k^{-1}$ by (11).

(2) Prediction: form $\mathbf{x}_{k+1}$, and calculate prediction $\hat{y}_{k+1}$ of y_{k+1} by (6); namely,

$$\begin{aligned} \hat{y}_{k+1} = {} & [K(\mathbf{x}_{k+1}, \mathbf{x}_{k-z+1}), \ldots, K(\mathbf{x}_{k+1}, \mathbf{x}_k)] \\ & \times \mathbf{A}_k^{-1}[y_{k-z+1}, \ldots, y_k]^{\mathrm{T}}. \end{aligned} \tag{19}$$

(3) Delete the oldest sample $(\mathbf{x}_s, y_s)$: calculate $\overline{\mathbf{A}}^{-1}$ by (18).

4. Performance Evaluation

In this section, the performance of the presented FORELM and FOKELM is verified via the time-varying nonlinear process identification simulations. Those simulations are designed from the aspects of accuracy, stability, and computation complexity of the proposed FORELM and FOKELM by comparison with the FOS-ELM, ReOS-ELM (i.e., SRELM or LS-IELM), and dynamic regression extreme learning machine (DR-ELM) [24]. DR-ELM also is a kind of online sequential RELM and is designed by Zhang and Wang using solution (5b) of RELM and the block matrix inverse formula.

All the performance evaluations were executed in MATLAB 7.0.1 environment running on Windows XP with Intel Core i3-3220 3.3 GHz CPU and 4 GB RAM.

Simulation 1. The unknown identified system is a modified version of the one addressed in [25], by changing the constant and the coefficients of variables, to form a time-varying system, as done in [11]:

$$y(k+1)=\begin{cases}\dfrac{y(k)}{1+y(k)^2}+u(k)^3 & k\le 100\\ \dfrac{2y(k)}{2+y(k)^2}+u(k)^3 & 100<k\le 300\\ \dfrac{y(k)}{1+2y(k)^2}+2u(k)^3 & 300<k.\end{cases} \tag{20}$$

The system (20) can be expressed as follows:

$$y(k)=f(\mathbf{x}(k)), \tag{21}$$

where $f(\cdot)$ is a nonlinear function and $\mathbf{x}(k)$ is the regression input data vector

$$\mathbf{x}(k) = \left[y(k-1),\ldots,y(k-n_y),u(k-n_d),\ldots,u(k-n_u)\right], \tag{22}$$

with n_d, n_u, and n_y being model structure parameters. Apply SLFN to approximate (20); accordingly $(\mathbf{x}(k), y(k))$ is the learning sample $(\mathbf{x}_k, y_k)$ of SLFN.

Denote $k_0 = z+\max(n_y,n_u)-n_d$, $k_1 = k_0+350$. The input is set as follows:

$$u(k)=\begin{cases}\operatorname{rand}(\cdot) & k\le k_0\\ \sin\left(\dfrac{2\pi(k-k_0)}{120}\right) & k_0<k\le k_1\\ \sin\left(\dfrac{2\pi(k-k_1)}{50}\right) & k_1<k,\end{cases} \tag{23}$$

where rand$(\cdot)$ generates random numbers which are uniformly distributed in the interval $(0,1)$.

In all experiments, the output of hidden node with respect to the input $\mathbf{x}$ of a SLFN in (1) is set to the sigmoidal additive function; that is, $G(\mathbf{a},b,\mathbf{x}) = 1/(1+\exp(\mathbf{a}\cdot\mathbf{x}+b))$, the components of $\mathbf{a}$; that is, the input weights and bias b are randomly chosen from the range $[-2,2]$. In FOKELM, the Gaussian kernel function is applied; namely, $K(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\|x_i-x_j\|^2/\sigma)$.

The root-mean-square error (RMSE) of prediction and the maximal absolute prediction error (MAPE) are regarded as measure indices of model accuracy and stability, respectively. Consider

$$\text{RMSE}=\sqrt{\frac{1}{t_2-t_1+1}\sum_{i=t_1}^{t_2}(\hat{y}_i-y_i)^2},\qquad \text{MAPE}=\max_{i=t_1,\ldots,t_2}\left|\hat{y}_i-y_i\right|, \tag{24}$$

where $t_1=k_0+1$, $t_2=650$. Simulation is carried on for 650 instances. In model (21), $n_d=1$, $n_u=2$, and $n_y=3$. Due to randomness of parameters $\mathbf{a}$, b, and $u(t)$ during initial stage $(k\le k_0)$, the results of simulation must possess variation. For each approach, the results are averaged over 5 trials.

ReOS-ELM does not discard any old sample; thus it has $z\to\infty$.

In offline modeling, the training samples set is fixed; thus one may search relative optimal values for model parameter of ELMs. Nevertheless, in online modeling for time-varying system, the training samples set is changing; it is difficult to choose optimal values for parameters in practice. Therefore, we set the same parameter of these ELMs with the same value manually; for example, their parameters C all are set to 250; then we compare their performances.

RMSE and MAPE of the proposed ELMs and other aforementioned ELMs are listed in Tables 1 and 2, respectively, and the corresponding running time (i.e., training time plus predicting time) of these various ELMs is given in Table 3.

From Tables 1–3, one can see the following results.

(1) RMSE and MAPE of FORELM are smaller than those of FOS-ELM with the same z and L values. The reason for this is that $(\mathbf{H}^{\mathrm{T}}\mathbf{H})$ in FOS-ELM may be (nearly) singular at some instances; thus $(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}$ calculated recursively is nonsense and unbelievable, and when $L\ge z$, FOS-ELM cannot work owing to its too large RMSE or MAPE. Accordingly, "×" represents nullification in Tables 1–3, whereas FORELM does not suffer from such a problem. In addition, RMSE and MAPE of FOKELM also are smaller than those of FOS-ELM with the same z values.

(2) RMSE of FORELM and FOKELM is smaller than that of ReOS-ELM with the same L and C when parameter z, namely, the length of sliding time windows, is set properly. The reason for this is that ReOS-ELM neglects timeliness of samples of the time-varying process and does not get rid of effects of old samples; contrarily, FORELM and FOKELM stress actions of the newest ones. It should be noticed that z is relevant to characteristics of the specific time-varying system and its inputs. In Table 1, when $z = 50$, 70, or 100, the effect is good.

(3) When z is fixed, with L increasing, RMSEs of these ELMs tend to decrease firstly. But changes are not obvious later.

(4) FORELM requires nearly the same time as FOS-ELM, but more time than ReOS-ELM. It is because both FORELM and FOS-ELM involve p incremental and

Table 1: RMSE comparison between the proposed ELMs and other ELMs on identification of process (20) with input (23).

	z	50	70	100	150	200	∞
$L = 15$	FOS-ELM ($p = 2$)	0.3474	0.0967	0.0731	0.0975	0.0817	—
	ReOS-ELM	—	—	—	—	—	0.0961
	DR-ELM	0.0681	0.0683	0.0712	0.0833	0.0671	—
	FORELM ($p = 2$)	**0.0850**	**0.0708**	**0.0680**	**0.0816**	**0.0718**	—
$L = 25$	FOS-ELM	0.9070	0.1808	0.1324	0.0821	0.0883	—
	ReOS-ELM	—	—	—	—	—	0.0804
	DR-ELM	0.0543	0.0670	0.0789	0.0635	0.0656	—
	FORELM	**0.0578**	**0.0697**	**0.0833**	**0.0653**	**0.0600**	—
$L = 50$	FOS-ELM	×	×	×	9.9541	0.4806	—
	ReOS-ELM	—	—	—	—	—	0.0529
	DR-ELM	0.0377	0.0351	0.0345	0.0425	0.0457	—
	FORELM	**0.0344**	**0.0320**	**0.0324**	**0.0457**	**0.0456**	—
$L = 100$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.0359
	DR-ELM	0.0298	0.0306	0.0308	0.0391	0.0393	—
	FORELM	**0.0312**	**0.0263**	**0.0288**	**0.0405**	**0.0378**	—
$L = 150$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.0351
	DR-ELM	0.0268	0.0270	0.0281	0.0417	0.0365	—
	FORELM	**0.0270**	**0.0276**	**0.0284**	**0.0378**	**0.0330**	—
$L = 200$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.0344
	DR-ELM	0.0259	0.0284	0.0272	0.0363	0.0351	—
	FORELM	**0.0263**	**0.0289**	**0.0274**	**0.0355**	**0.0325**	—
$L = 400$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.0296
	DR-ELM	0.0240	0.0255	0.0249	0.0362	0.0327	—
	FORELM	0.0231	0.0248	0.0256	0.0372	0.0310	—
$L = 800$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.0292
	DR-ELM	0.0233	0.0253	0.0270	0.0374	0.0320	—
	FORELM	0.0223	0.0256	0.0252	0.0407	0.0318	—
$L = 1000$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.0298
	DR-ELM	0.0234	0.0263	0.0278	0.0418	0.0323	—
	FORELM	**0.0232**	**0.0238**	**0.0259**	**0.0372**	**0.0310**	**—**
	FOKELM	**0.0236**	**0.0251**	**0.0267**	**0.0386**	**0.0331**	—

"—" represents nondefinition or inexistence in the case.
"×" represents nullification owing to the too large RMSE or MAPE.

decremental learning procedures, but ReOS-ELM does one incremental learning procedure only.

(5) Both FORELM and DR-ELM use regularization trick, so they should obtain the same or similar prediction effect theoretically, and Tables 1 and 2 also show they have similar simulation results statistically. However, their time efficiencies are different. From Table 3, it can be seen that, for the case where L is small or $L \leq z$, FORELM takes less time than DR-ELM. Thus, when L is small and modeling speed is preferred to accuracy, one may try to employ FORELM.

(6) When z is fixed, if L is large enough, there are no significant differences between RMSE of FORELM and DR-ELM and RMSE of FOKELM with appropriate parameter values. In Table 3, it is obvious that, with L increasing, DR-ELM costs more and more time. But time cost by FOKELM is irrelevant with L. According to the procedures of DR-ELM and FOKELM, if L is large enough to make calculating $\mathbf{h}(\mathbf{x}_i)\mathbf{h}(\mathbf{x}_j)^{\mathrm{T}}$ more

Table 2: MAPE comparison between the proposed ELMs and other ELMs on identification of process (20) with input (23).

	z	50	70	100	150	200	∞
$L = 15$	FOS-ELM ($p = 2$)	5.8327	0.6123	0.5415	1.0993	0.7288	—
	ReOS-ELM	—	—	—	—	—	0.3236
	DR-ELM	0.3412	0.4318	0.3440	0.7437	0.4007	—
	FORELM ($p = 2$)	**0.2689**	**0.3639**	**0.3135**	**0.8583**	**0.4661**	—
$L = 25$	FOS-ELM	5.0473	1.7010	2.4941	0.9537	0.9272	—
	ReOS-ELM	—	—	—	—	—	0.4773
	DR-ELM	0.3418	0.3158	0.4577	0.5680	0.3127	—
	FORELM	**0.3391**	**0.2568**	**0.3834**	**0.6228**	**0.2908**	—
$L = 50$	FOS-ELM	×	×	×	209.8850	3.9992	—
	ReOS-ELM	—	—	—	—	—	0.2551
	DR-ELM	0.3876	0.3109	0.3808	0.5486	0.3029	—
	FORELM	**0.3478**	**0.2960**	**0.3145**	**0.5504**	**0.3514**	—
$L = 100$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.2622
	DR-ELM	0.2473	0.2748	0.2163	0.5471	0.3381	—
	FORELM	**0.3229**	**0.2506**	**0.2074**	**0.5501**	**0.2936**	—
$L = 150$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.3360
	DR-ELM	0.2725	0.2365	0.2133	0.5439	0.3071	—
	FORELM	**0.2713**	**0.2418**	**0.2050**	**0.5432**	**0.3133**	—
$L = 200$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.2687
	DR-ELM	0.2539	0.2454	0.2069	0.5443	0.3161	—
	FORELM	**0.2643**	**0.2370**	**0.2029**	**0.5443**	**0.3148**	—
$L = 400$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.2657
	DR-ELM	0.2520	0.2456	0.2053	0.5420	0.3260	—
	FORELM	**0.2277**	**0.2382**	**0.2285**	**0.5430**	**0.3208**	—
$L = 800$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.3784
	DR-ELM	0.2388	0.2401	0.3112	0.5415	0.3225	—
	FORELM	0.2128	0.2325	0.3006	0.5416	0.3247	—
$L = 1000$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.4179
	DR-ELM	0.2230	0.2550	0.3412	0.5431	0.3283	—
	FORELM	**0.2228**	**0.2525**	**0.3347**	**0.5412**	**0.3293**	**—**
FOKELM		**0.2397**	**0.2345**	**0.3125**	**0.5421**	**0.3178**	—

complex than calculating $K(\mathbf{x}_i, \mathbf{x}_j)$, FOKELM will take less time than DR-ELM.

To intuitively observe and compare the accuracy and stability of these ELMs with the same $\mathbf{a}$, b values and initial $u(k)$ $(k \le k_0)$ signal, absolute prediction error (APE) curves of one trial of every approach ($L = 25$, $z = 70$) are shown in Figure 1. Clearly, Figure 1(a) shows that, at a few instances, prediction errors of FOS-ELM are much greater, although prediction errors are very small at other instances; thus FOS-ELM is unstable. Comparing Figures 1(b), 1(c), 1(d), and 1(e), we can see that, at most instances, prediction errors of DR-ELM, FORELM, and FOKELM are smaller than those of ReOS-ELM, and prediction effect of FORELM is similar to that of DR-ELM.

On the whole, FORELM and FOKELM have higher accuracy than FOS-ELM and ReOS-ELM.

Simulation 2. In this subsection, the proposed methods are tested on modeling for a second-order bioreactor process described by the following differential equations [26]:

$$\dot{c}_1(t) = -c_1(t)\,u(t) + c_1(t)\left(1 - c_2(t)\right)e^{c_2(t)/\gamma},$$
$$\dot{c}_2(t) = -c_2(t)\,u(t) + c_1(t)\left(1 - c_2(t)\right)e^{c_2(t)/\gamma}\,\frac{1+\beta}{1+\beta-c_2(t)}, \tag{25}$$

where $c_1(t)$ is the cell concentration that is considered as the output of the process ($y(t) = c_1(t)$), $c_2(t)$ is the amount of nutrients per unit volume, and $u(t)$ represents the flow rate as the control input (the excitation signal for modeling); the $c_1(t)$ and $c_2(t)$ can take values between zero and one, and $u(t)$ is allowed a magnitude in interval $[0, 2]$.

Table 3: Running time (second) comparison between the proposed ELMs and other ELMs on identification of process (20) with input (23).

	z	50	70	100	150	200	∞
$L = 15$	FOS-ELM ($p = 2$)	0.0620	0.0620	0.0621	0.0621	0.0622	—
	ReOS-ELM	—	—	—	—	—	0.0160
	DR-ELM	0.2650	0.3590	0.5781	1.0790	1.6102	—
	FORELM ($p = 2$)	**0.0620**	**0.0620**	**0.0620**	**0.0621**	**0.0622**	—
$L = 25$	FOS-ELM	0.0621	0.0621	0.0621	0.0623	0.0623	—
	ReOS-ELM	—	—	—	—	—	0.0167
	DR-ELM	0.2500	0.3750	0.6094	1.0630	1.6250	—
	FORELM	**0.0621**	**0.0621**	**0.0621**	**0.0622**	**0.0622**	—
$L = 50$	FOS-ELM	×	×	×	0.1250	0.1250	—
	ReOS-ELM	—	—	—	—	—	0.0470
	DR-ELM	0.2650	0.3906	0.6250	1.1100	1.6560	—
	FORELM	**0.1400**	**0.1562**	**0.1560**	**0.1250**	**0.1250**	—
$L = 100$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.1250
	DR-ELM	0.2813	0.4219	0.6719	1.2031	1.7380	—
	FORELM	**0.3750**	**0.4063**	**0.3440**	**0.3440**	**0.3280**	—
$L = 150$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.2344
	DR-ELM	0.3125	0.4540	0.7500	1.2970	1.9220	—
	FORELM	**0.7810**	**0.8003**	**0.6560**	**0.6720**	**0.6570**	—
$L = 200$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	0.3901
	DR-ELM	0.3290	0.5160	0.7810	1.3440	1.9690	—
	FORELM	**1.2702**	**1.2750**	**1.1250**	**1.0320**	**0.9375**	—
$L = 400$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	1.7350
	DR-ELM	0.3906	0.5781	0.8750	1.4680	2.2820	—
	FORELM	6.6410	6.4530	6.2030	5.6090	5.1719	—
$L = 800$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	8.7190
	DR-ELM	0.5470	0.7970	1.2660	2.0938	2.9840	—
	FORELM	32.7820	31.6720	30.1400	27.5780	27.7500	—
$L = 1000$	FOS-ELM	×	×	×	×	×	—
	ReOS-ELM	—	—	—	—	—	13.6094
	DR-ELM	0.6250	1.0000	1.4530	2.4380	3.4060	—
	FORELM	**51.4530**	**49.3440**	**46.8750**	**43.1880**	**38.9060**	**—**
FOKELM		**0.2811**	**0.4364**	**0.6951**	**1.1720**	**1.7813**	—

In the simulation, with the growth rate parameter $\beta = 0.02$, the nutrient inhibition parameter γ is considered as the time-varying parameter; that is,

$$\gamma = \begin{cases} 0.3 & 0\,\mathrm{s} < t \le 20\,\mathrm{s} \\ 0.48 & 20\,\mathrm{s} < t \le 60\,\mathrm{s} \\ 0.6 & 60\,\mathrm{s} < t. \end{cases} \tag{26}$$

Let T_s indicate sampling interval. Denote $t_0 = (z + \max(n_y, n_u) - n_d)T_s$, $t_1 = t_0 + 350T_s$. The input is set below:

$$u(t) = \begin{cases} 0.5\,\mathrm{rand}(\cdot) + 0.75 & t \le t_0 \\ 0.25 \sin\left(\dfrac{2\pi (t - t_0)}{24}\right) + 1 & t_0 < t \le t_1 \\ 0.25 \sin\left(\dfrac{2\pi (t - t_1)}{10}\right) + 1 & t_1 < t, \end{cases} \tag{27}$$

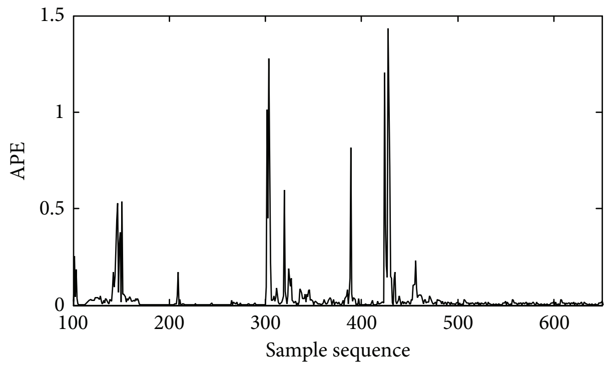
where, at every sampling instance, rand(·) generates random numbers which are uniformly distributed in the interval $(0, 1)$.

Set $T_s = 0.2$ s. With the same a, b, and initial $u(t)$, APE curves of every approach ($L = 20$, $z = 100$) for one trial are drawn in Figure 2. Clearly, on the whole, APE curves of FORELM and FOKELM are smaller than those of FOS-ELM and ReOS-ELM, and FORELM has nearly the same prediction effect as DR-ELM. Further, RMSE of FOS-ELM, ReOS-ELM, DR-ELM, FORELM, and FOKELM are 0.096241, 0.012203, 0.007439, 0.007619, and 0.007102, respectively.

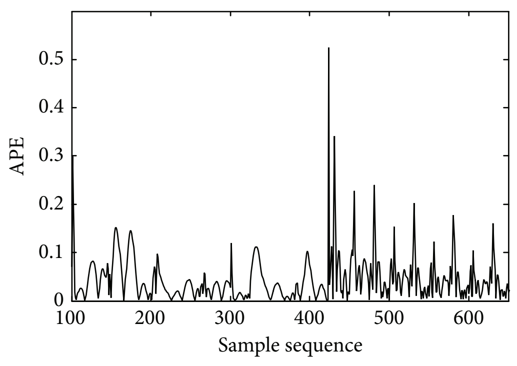
Through many comparative trials, we may attain the same results as the ones in Simulation 1.

5. Conclusions

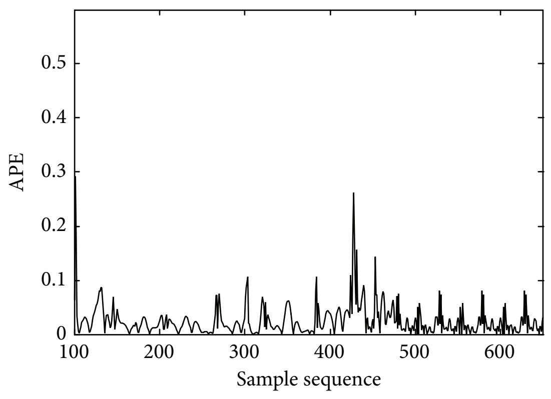
ReOS-ELM (i.e., SRELM or LS-IELM) can yield good generalization models and will not suffer from matrix singularity



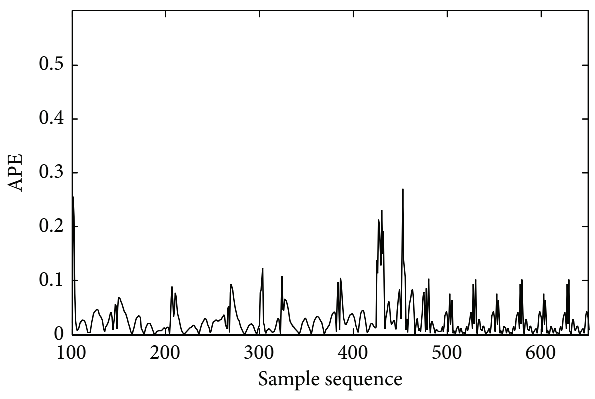
(a) APE curves of FOS-ELM



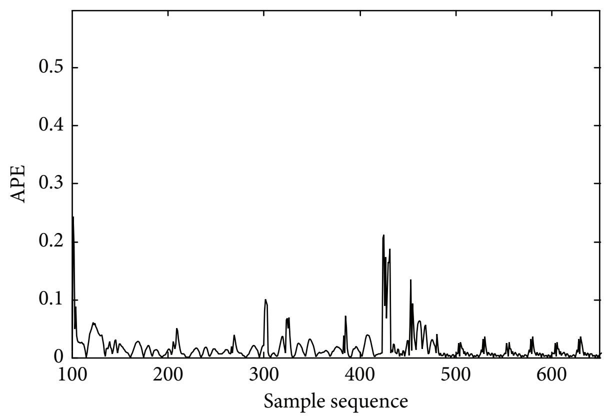
(b) APE curves of ReOS-ELM



(c) APE curves of DR-ELM



(d) APE curves of FORELM



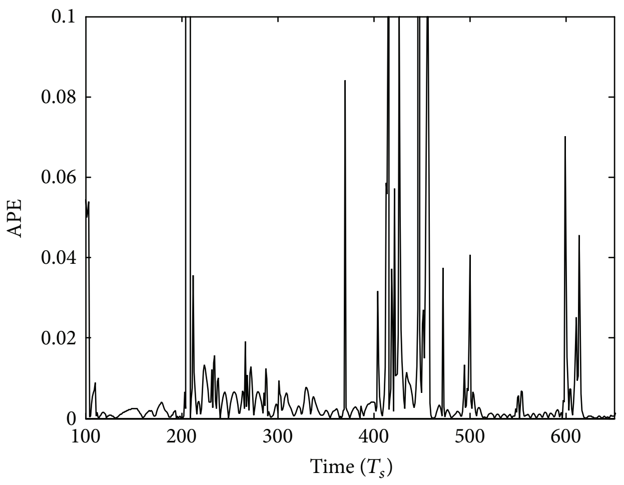
(e) APE curves of FOKELM

Figure 1: APE curves of ELMs on process (20) with input (23).

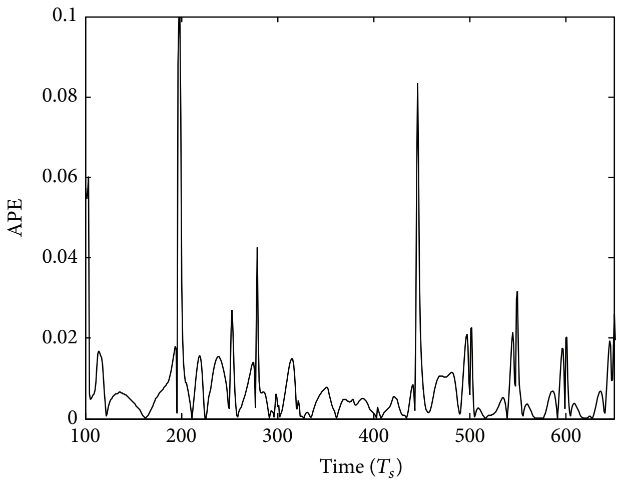
or ill-posed problems, but it is unsuitable in time-varying applications. On the other hand, FOS-ELM, thanks to its forgetting mechanism, can reflect the timeliness of data and train SLFN in nonstationary environments, but it may encounter the matrix singularity problem and run unstably.

In the paper, the forgetting mechanism is incorporated to ReOS-ELM, and we obtain FORELM which blends advantages of ReOS-ELM and FOS-ELM. In addition, the forgetting mechanism also is added to KB-IELM; consequently, FOKELM is obtained, which can overcome matrix expansion problem of KB-IELM.

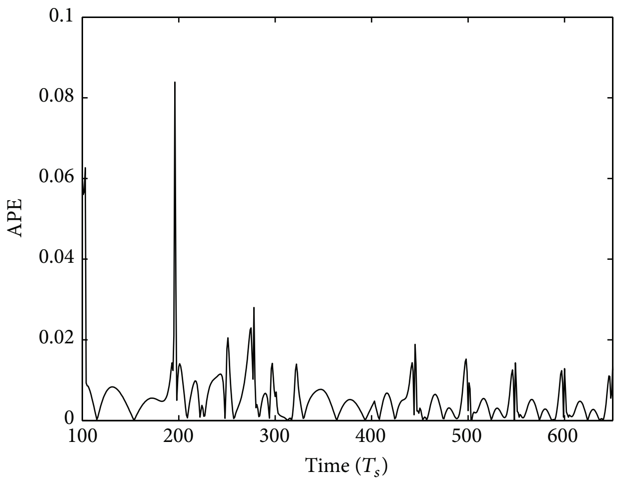
Performance comparison between the proposed ELMs and other ELMs was carried out on identification of time-varying systems in the aspects of accuracy, stability, and computational complexity. The experimental results show that FORELM and FOKELM have better stability than FOS-ELM and have higher accuracy than ReOS-ELM in nonstationary environments statistically. When the number L of hidden nodes is small or $L \leq$ the length z of sliding time windows, FORELM has time efficiency superiority over DR-ELM. On the other hand, if L is large enough, FOKELM will be faster than DR-ELM.



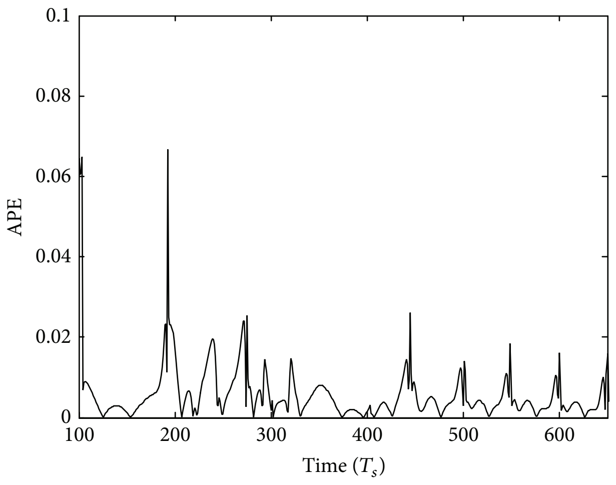
(a) APE curves of FOS-ELM



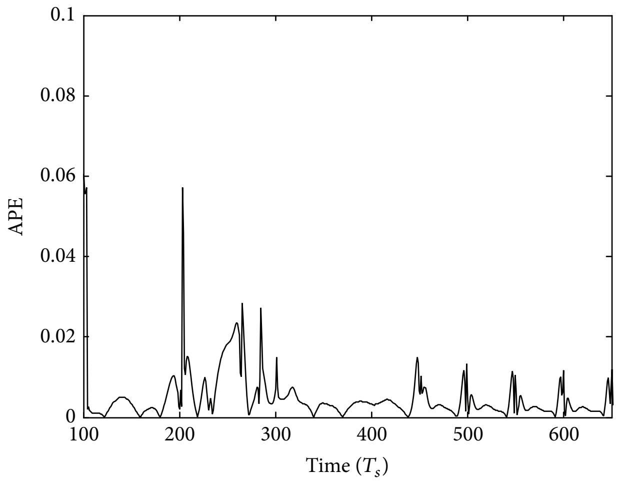
(b) APE curves of ReOS-ELM



(c) APE curves of DR-ELM



(d) APE curves of FORELM



(e) APE curves of FOKELM

Figure 2: APE curves of ELMs on process (25) with time-varying γ in (26).

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The work is supported by the Hunan Provincial Science and Technology Foundation of China (2011FJ6033).

References

[1] J. Park and I. W. Sandberg, "Universal approximation using radial basis function networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, 1991.

[2] G. B. Huang, Y. Q. Chen, and H. A. Babri, "Classification ability of single hidden layer feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 799–801, 2000.

[3] S. Ferrari and R. F. Stengel, "Smooth function approximation using neural networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 24–38, 2005.

[4] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 985–990, July 2004.

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

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

[7] J. S. Lim, "Partitioned online sequential extreme learning machine for large ordered system modeling," *Neurocomputing*, vol. 102, pp. 59–64, 2013.

[8] J. S. Lim, S. Lee, and H. S. Pang, "Low complexity adaptive forgetting factor for online sequential extreme learning machine (OS-ELM) for application to nonstationary system estimations," *Neural Computing and Applications*, vol. 22, no. 3-4, pp. 569–576, 2013.

[9] Y. Gu, J. F. Liu, Y. Q. Chen, X. L. Jiang, and H. C. Yu, "TOSELM : timeliness online sequential extreme learning machine," *Neurocomputing*, vol. 128, no. 27, pp. 119–127, 2014.

[10] J. W. Zhao, Z. H. Wang, and D. S. Park, "Online sequential extreme learning machine with forgetting mechanism," *Neurocomputing*, vol. 87, no. 15, pp. 79–89, 2012.

[11] X. Zhang and H. L. Wang, "Fixed-memory extreme learning machine and its applications," *Control and Decision*, vol. 27, no. 8, pp. 1206–1210, 2012.

[12] W. Y. Deng, Q. H. Zheng, and L. Chen, "Regularized extreme learning machine," in *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM '09)*, pp. 389–395, April 2009.

[13] H. T. Huynh and Y. Won, "Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks," *Pattern Recognition Letters*, vol. 32, no. 14, pp. 1930–1935, 2011.

[14] X. Zhang and H. L. Wang, "Time series prediction based on sequential regularized extreme learning machine and its application," *Acta Aeronautica et Astronautica Sinica*, vol. 32, no. 7, pp. 1302–1308, 2011.

[15] G. B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[16] G. B. Huang, H. M. Zhou, X. J. Ding, and R. Zhang, "Extreme learning ," *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[17] V. N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, New York, NY, USA, 1998.

[18] L. Guo, J. H. Hao, and M. Liu, "An incremental extreme learning machine for online sequential learning problems," *Neurocomputing*, vol. 128, no. 27, pp. 50–58, 2014.

[19] G. B. Huang, L. Chen, and C. K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[20] G. B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16–18, pp. 3056–3062, 2007.

[21] G. B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16–18, pp. 3460–3468, 2008.

[22] X. Zhang and H. L. Wang, "Incremental regularized extreme learning machine based on Cholesky factorization and its application to time series prediction," *Acta Physica Sinica*, vol. 60, no. 11, Article ID 110201, 2011.

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

[24] X. Zhang and H. L. Wang, "Dynamic regression extreme learning machine and its application to small-sample time series prediction," *Information and Control*, vol. 40, no. 5, pp. 704–709, 2011.

[25] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[26] M. O. Efe, E. Abadoglu, and O. Kaynak, "A novel analysis and design of a neural network assisted nonlinear controller for a bioreactor," *International Journal of Robust and Nonlinear Control*, vol. 9, no. 11, pp. 799–815, 1999.