Research Article

Online Sequential Prediction for Nonstationary Time Series with New Weight-Setting Strategy Using Extreme Learning Machine

Wentao Mao,¹ Jinwan Wang,¹ Liyun Wang,¹ and Mei Tian²

¹College of Computer and Information Engineering, Henan Normal University, Henan, Xinxiang 453007, China

²Management Institute, Xinxiang Medical University, Henan, Xinxiang 453003, China

Correspondence should be addressed to Wentao Mao; maowt.mail@gmail.com

Received 21 August 2014; Accepted 12 October 2014

Academic Editor: Amaury Lendasse

Copyright © 2015 Wentao Mao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Accurate and fast prediction of nonstationary time series is challenging and of great interest in both practical and academic areas. In this paper, an online sequential extreme learning machine with new weighted strategy is proposed for nonstationary time series prediction. First, a new leave-one-out (LOO) cross-validation error estimation for online sequential data is proposed based on inversion of block matrix. Second, a new weighted strategy based on the proposed LOO error estimation is proposed. This strategy ranks the samples' importance by means of the LOO error of each new added sample and then assigns various weights. Performance comparisons of the proposed method with other existing algorithms are presented based on chaotic and real-world nonstationary time series data. The results show that the proposed method outperforms the classical ELM and OS-ELM in terms of generalization performance and numerical stability.

1. Introduction

Time series prediction is generally playing an important role in many engineering fields, for example, dynamic mechanics, weather diagnostics, and so on. The key goal of time series prediction is to mine the inner regular patterns in time data in order to predict future data effectively [1]. Many traditional methods such as AR, ARMA, and ARIMA are well applied to solving stationary time series prediction. However, in practical applications, time series is almost nonstationary, which restricts the stationary methods above. From Takens' phase space delay reconstructing theory [2], this kind of data generally needs to reconstruct the phase space via delay coordinate at first. For example, in chaotic time series the m -dimensional vector is defined as follows:

$$x(t-1) = [x(t-\tau), x(t-2\tau), \dots, x(t-m\tau)], \quad (1)$$

where m is embedding dimension and τ is delay constant. The prediction model can be described as $x(t) = f(x(t-1))$, where f is a nonlinear map. From this reconstruction,

the time correlation is transformed to spatial correlation. Then support vector machines (SVMs) [3, 4], neural networks (NNs) [5, 6], and other machine learning methods [7, 8] are successfully introduced to approximate the spatial correlation in nonstationary time series data.

Generally speaking, there are two main challenges for predicting nonstationary time series effectively. One is how to choose a proper baseline algorithm which should be computationally inexpensive and accurate enough. Another is how to distinguish the importance of different samples in time series. Different from SVMs and NNs, extreme learning machine (ELM), introduced by Huang et al. [9], has shown its very high learning speed and good generalization performance in solving many problems of regression estimate and pattern recognition [10, 11]. As a sequential modification of ELM, online sequential ELM (OS-ELM) proposed by Liang et al. [12] can learn data one-by-one or chunk-by-chunk. In many applications such as time-series forecasting, OS-ELMs also show good generalization at extremely fast learning speed. Therefore, OS-ELM is a proper solution for the first

challenge. Many researches were devoted to solve the second challenge. As recent data usually carry more important information than the distant past data, a typical and effective method is weight-setting. Lin and Wang [13] held the first sample in time series with the lowest importance while the most recent sample with the highest importance and then assigned fuzzy memberships to every sample. Tay and Cao [14] used exponential function to calculate every sample's importance in financial time series prediction. Very different from these stationary weight-setting strategy, Mao et al. [15] established a heuristic algorithm to dynamically choose the optimal weights. Bao et al. [16] solved this problem from multi-input multioutput perspective. He regarded the time samples as multiple outputs in a time slot, and utilized multidimensional SVM to establish model. Considering ELM, Wang and Han [17] introduced kernel trick on OS-ELM for nonstationary time series. Its essential idea is to transform spatial space for better approximation. Grigorievskiy et al. [18] used optimally-pruned ELM to tackle long-term time series prediction and obtained more comparable results than with SVM.

However, although ELM-based methods mentioned above work well in time series prediction [19], it still does not yet successfully solve the second challenge, that is, to distinguish different samples' significance. Specifically speaking, as a sample is added sequentially, it does not seem clear whether this sample is the most important or is even the newest. In this scenario, the inner structure hidden in time series data will determine the samples' significance, especially in nonstationary setting. In other words, it could not guarantee the new added sample to be most valuable for prediction. Therefore, to solve this problem, this paper firstly develops a new leave-one-out (LOO) cross-validation error estimation for OS-ELM aiming at time series prediction. Based on inversion of block matrix, this LOO estimation is fast enough for time series data. As proved by many theoretical works [20], LOO error using PRESS statistics is approximately unbiased and has been successfully applied to ELM with Tikhonov regularization [21]. To our best knowledge, this LOO error estimation is the first attempt to evaluate the generalization performance of OS-ELM on time series data. Moreover, this paper utilizes this LOO error estimation of each new added sample to measure its importance. Obeying this weight-setting strategy, this paper then proposes a new weighted learning method for OS-ELM. The short version of this paper has been published in the proceedings of 5th International Conference on Extreme Learning Machine (ELM 2014). Experimental results on chaotic and real-life time series data demonstrate the proposed method outperforms the traditional ELMs in generalization performance and numerical stable.

The paper is organized as follows. In Section 2, a brief review on OS-ELM and LOO cross-validation estimation is provided. In Section 3, we describe the LOO error estimation and the weighted learning algorithm of OS-ELM on time series data. Section 4 is devoted to computer experiments on two different types of time series data sets, followed by a conclusion of the paper in Section 5.

2. Brief Review

As the theoretical foundations of ELM, [22] studied the learning performance of SLFN on small-size data set and found that SLFN with at most N hidden neurons can learn N distinct samples with zero errors by adopting any bounded nonlinear activation function. Then, based on this concept, Huang et al. [9] pointed out that ELM can analytically determine the output weights by a simple matrix inversion procedure as soon as the input weights and hidden layer biases are generated randomly and then obtain good generalization performance with very high learning speed. Here a brief summary of ELM is provided.

Given a set of i.i.d. training samples $\{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)\} \subset \mathbf{R}^d \times \mathbf{R}^n$, standard SLFNs with $\tilde{N}$ hidden nodes are mathematically formulated as [9]:

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(\mathbf{x}_j) = \sum_{i=1}^{\tilde{N}} \beta_i g_i(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{o}_j, \quad j = 1, \dots, N, \quad (2)$$

where $g(x)$ is activation function, $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{id}]^T$ is input weight vector connecting input nodes and the i th hidden node, $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{in}]^T$ is the output weight vector connecting output nodes and the i th hidden node, and b_i is bias of the i th hidden node. Huang et al. [9] has rigorously proved that, then, for N arbitrary distinct samples and any $(\mathbf{w}_i, b_i)$ randomly chosen from $\mathbf{R}^d \times \mathbf{R}$ according to any continuous probability distribution, the hidden layer output matrix $\mathbf{H}$ of a standard SLFN with N hidden nodes and is invertible and $\|\mathbf{H}\beta - \mathbf{T}\| = 0$ with probability one if the activation function $g: \mathbf{R} \mapsto \mathbf{R}$ is infinitely differentiable in any interval. Then, given $(\mathbf{w}_i, b_i)$, training a SLFN equals finding a least-squares solution of the following equation [9]:

$$\mathbf{H}\beta = \mathbf{T}, \quad (3)$$

where

$$\mathbf{H}(\mathbf{w}_1, \dots, \mathbf{w}_{\tilde{N}}, b_1, \dots, b_{\tilde{N}}, \mathbf{x}_1, \dots, \mathbf{x}_{\tilde{N}}) = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_1 + b_{\tilde{N}}) \\ \vdots & & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_{\tilde{N}} + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_{\tilde{N}} + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}, \quad (4)$$

$$\beta = [\beta_1, \dots, \beta_{\tilde{N}}]^T,$$

$$\mathbf{T} = [\mathbf{y}_1, \dots, \mathbf{y}_{\tilde{N}}]^T.$$

Considering most cases in which $\tilde{N} \ll N$, β cannot be computed through the direct matrix inversion. Therefore, the smallest norm least-squares solution of (3) is calculated as follows:

$$\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}, \quad (5)$$

where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$. Based on the analysis above, Huang et al. [9] proposed ELM whose framework can be stated as follows.

Step 1. Randomly generate input weight and bias $(\mathbf{w}_i, b_i)$, $i = 1, \dots, \tilde{N}$.

Step 2. Compute the hidden layer output matrix $\mathbf{H}$.

Step 3. Compute the output weight $\hat{\boldsymbol{\beta}} = \mathbf{H}^\dagger \mathbf{T}$.

Therefore, the output of SLFN can be calculated by $(\mathbf{w}_i, b_i)$ and $\hat{\boldsymbol{\beta}}$:

$$f(\mathbf{x}_j) = \sum_{i=1}^{\tilde{N}} \hat{\beta}_i g_i(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \hat{\boldsymbol{\beta}} \cdot \mathbf{h}(\mathbf{x}_j). \quad (6)$$

Like ELM, all the hidden node parameters in OS-ELM are randomly generated, and the output weights are analytically determined based on the sequentially arrived data. OS-ELM process is divided into two steps: initialization phase and sequential learning phase [12].

Step 1. Initialization phase: choose a small chunk $M_0 = \{(\mathbf{x}_i, t_i), i = 1, 2, \dots, N_0\}$ of initial training data, where $N_0 \geq \tilde{N}$.

- (1) Randomly generate the input weight $\mathbf{w}_i$ and bias b_i , $i = 1, 2, \dots, \tilde{N}$. Calculate the initial hidden layer output matrix $\mathbf{H}_0$:

$$\mathbf{H}_0 = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \mathbf{h}(\mathbf{x}_2) \\ \vdots \\ \mathbf{h}(\mathbf{x}_{N_0}) \end{bmatrix} \quad (7)$$

$$= \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_1 + b_{\tilde{N}}) \\ g(\mathbf{w}_1 \cdot \mathbf{x}_2 + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_2 + b_{\tilde{N}}) \\ \vdots & & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_{N_0} + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_{N_0} + b_{\tilde{N}}) \end{bmatrix}_{N_0 \times \tilde{N}}.$$

- (2) Calculate the output weight vector:

$$\boldsymbol{\beta}^0 = \mathbf{D}_0 \mathbf{H}_0^T \mathbf{T}_0, \quad (8)$$

where $\mathbf{D}_0 = (\mathbf{H}_0^T \mathbf{H}_0)^{-1}$, $\mathbf{T}_0 = [t_1, t_2, \dots, t_{N_0}]^T$.

- (3) Set $k = 0$.

Step 2. Sequential learning phase.

- (1) Learn the $(k + 1)$ th training data: $d_{k+1} = (\mathbf{x}_{N_0+k+1}, t_{N_0+k+1})$.
- (2) Calculate the partial hidden layer output matrix:

$$\mathbf{H}_{k+1} = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_{N_0+k+1} + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_{N_0+k+1} + b_{\tilde{N}}) \end{bmatrix}_{1 \times \tilde{N}}. \quad (9)$$

Set $\mathbf{T}_{k+1} = [t_{N_0+k+1}]^T$.

- (3) Calculate the output weight vector:

$$\mathbf{D}_{k+1} = \mathbf{D}_k - \mathbf{D}_k \mathbf{H}_{k+1}^T (\mathbf{I} + \mathbf{H}_{k+1} \mathbf{D}_k \mathbf{H}_{k+1}^T)^{-1} \mathbf{H}_{k+1} \mathbf{D}_k \quad (10)$$

$$\boldsymbol{\beta}^{k+1} = \boldsymbol{\beta}^k + \mathbf{D}_{k+1} \mathbf{H}_{k+1}^T (\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\beta}^k).$$

- (4) Set $k = k + 1$. Go to Step 2(1).

The generalization ability of ELM has been analyzed by many researchers. Lan et al. [23] added a refinement stage that used leave-one-out (LOO) error to evaluate the neurons significance in each backward step. Feng et al. [24] presented a fast LOO error estimation for regularized ELM. From the incremental learning point of view, Feng et al. [24] proposed an error minimized extreme learning machine which measured the residual error caused by adding a new added hidden node in an incremental manner. We highly recommend the following work. As for ELM, Liu et al. [25] derived a fast LOO error estimation of ELM. The generalization error in i th LOO iteration can be expressed as follows:

$$r_i = t_i - f_i(\mathbf{x}_i) = \frac{t_i - \mathbf{H} \mathbf{x}_i \mathbf{H}^\dagger \mathbf{T}}{1 - (\mathbf{H} \mathbf{x}_i \mathbf{H}^\dagger)_i}, \quad (11)$$

where $(\cdot)_i$ means the i th element, $\mathbf{H}$ is hidden layer matrix, and $\mathbf{H}_{\mathbf{x}_i}$ means the row about the sample $\mathbf{x}_i$ in $\mathbf{H}$.

Liu et al. [25] have shown that the proposed algorithm can accurately calculate the LOO error and can avoid the N times observable model training process of the original cross-validation method. By the simulation experiment of artificial and real data sets, it has been verified that the LOO cross-validation algorithm based on ELM is efficient and has good generalization performance.

3. OS-ELM with LOO Weighted Strategy

As shown above, in the training process of the classic OS-ELM algorithm, all samples are equally treated. As long as a new sample is arriving, the network weight will be updated. This rigid weight updating mechanism lacks adjustment flexibility according to the actual situation. Moreover, it tends to increase the unnecessary computation.

To improve the generalization ability of OS-ELM while maintaining model's simplicity, this paper improves this rigid weight updating mechanism of traditional OS-ELM effectively via adopting dynamic weighted strategy. This strategy determines each sample's importance according to its LOO error estimation in online scenario. Consequently, a new OS-ELM based on online LOO cross-validation weight-setting strategy (LW-OS-ELM) is proposed.

3.1. LOO Error Estimation of ELM. As discussed in Section 2, the fast LOO error estimation of ELM proposed by Feng et al. [24] derived that the generalization error in i th LOO iteration can be expressed as follows:

$$r_i = t_i - f_i(\mathbf{x}_i) = \frac{t_i - \mathbf{H} \mathbf{x}_i \mathbf{H}^\dagger \mathbf{T}}{1 - (\mathbf{H} \mathbf{x}_i \mathbf{H}^\dagger)_i}. \quad (12)$$

Obviously, (12) works mainly on offline learning setting rather than online sequential scenario. The key reason is that $\mathbf{H}^+$ cannot be updated in online stage. Considering the sequential reaching of the sample in online setting, (12) can be extended to online sequential scenario, and it provides a channel to calculate the LOO error of each sample. The key is calculating $\mathbf{H}^+$ from (12) in online manner. However, the time complexity of matrix inversion is $O(n^3)$, where n is the number of samples. Therefore, the modeling time will be significantly increased along with the number of training patterns. To avoid complex calculation and make the established model simple, we adopt the idea of block matrix inversion, which transforms the complex calculation into linear operation, for decreasing the computation greatly.

As pointed out by many theoretical researches, LOO error is almost unbiased estimation of true generalization performance. Once a sample's LOO error is smaller, this sample's contribution in the decision model is greater. In order to highlight the samples' contribution and ensure the generalization of models, we set the corresponding weights of each sample according to the value of LOO error in the process of online. At the same time, to ensure the simplicity of the model, the oldest sample, which has the furthest distance from the current moment, is eliminated. Namely, the samples cost is zero. To avoid complex calculation and make the established model simple, we follow the idea of block matrix inversion [26], which transforms the complex calculation into linear operation, for decreasing the computation greatly in online learning stage. Thus, a new kind of extreme learning machine based on online leave-one-out cross-validation is put forward, as in the following sections.

3.2. Initial Stage of Training. Suppose there are N training samples $(\mathbf{x}_{i+1}, t_{i+1}), \dots, (\mathbf{x}_{i+N}, t_{i+N})$. The hidden layer output matrix is $\mathbf{H}_i = [\mathbf{h}_{i+1}^T \cdots \mathbf{h}_{i+N}^T]^T$, and the output vector is $\mathbf{T}_i = [t_{i+1} \cdots t_{i+N}]^T$, calculating the output weight vector:

$$\beta_i = \mathbf{H}_i^+ \mathbf{T}_i, \quad (13)$$

where

$$\mathbf{H}_i^+ = \mathbf{H}_i^T (\mathbf{H}_i \mathbf{H}_i^T)^{-1}. \quad (14)$$

Let $\mathbf{A}_i = \mathbf{H}_i \mathbf{H}_i^T$; (14) can be rewritten as $\mathbf{H}_i^+ = \mathbf{H}_i^T \mathbf{A}_i^{-1}$.

3.3. Add New Sample. Add the new arrived sample $(\mathbf{x}_{i+N+1}, t_{i+N+1})$ into training set. The output vector becomes $\mathbf{T}_{i+1} = [t_{i+1} \cdots t_{i+N} \ t_{i+N+1}]^T = [\mathbf{T}_i^T \ t_{i+N+1}]^T$, and the hidden layer matrix becomes $\mathbf{H}_{i+1} = [\mathbf{H}_i^T \ \mathbf{h}_{i+N+1}^T]^T$. Then we have

$$\mathbf{H}_{i+1}^+ = \mathbf{H}_{i+1}^T (\mathbf{H}_{i+1} \mathbf{H}_{i+1}^T)^{-1}. \quad (15)$$

Let $\mathbf{A}_{i+1} = \mathbf{H}_{i+1} \mathbf{H}_{i+1}^T$; then

$$\mathbf{H}_{i+1}^+ = \mathbf{H}_{i+1}^T \mathbf{A}_{i+1}^{-1}, \quad (16)$$

because

$$\begin{aligned} \mathbf{A}_{i+1} &= \mathbf{H}_{i+1} \mathbf{H}_{i+1}^T = [\mathbf{H}_i^T \ \mathbf{h}_{i+N+1}^T]^T [\mathbf{H}_i^T \ \mathbf{h}_{i+N+1}^T] \\ &= \begin{bmatrix} \mathbf{H}_i \mathbf{H}_i^T & \mathbf{H}_i \mathbf{h}_{i+N+1}^T \\ \mathbf{h}_{i+N+1} \mathbf{H}_i^T & \mathbf{h}_{i+N+1} \mathbf{h}_{i+N+1}^T \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A}_i & \mathbf{H}_i \mathbf{h}_{i+N+1}^T \\ \mathbf{h}_{i+N+1} \mathbf{H}_i^T & \mathbf{h}_{i+N+1} \mathbf{h}_{i+N+1}^T \end{bmatrix}. \end{aligned} \quad (17)$$

For (17), according to Block matrix inversion, we have

$$\mathbf{A}_{i+1}^{-1} = \begin{bmatrix} \mathbf{A}_i^{-1} \\ \mathbf{0}^T \end{bmatrix} + \frac{1}{B} \begin{bmatrix} \mathbf{A}_i^{-1} \mathbf{C} \\ -1 \end{bmatrix} [\mathbf{C}^T \mathbf{A}_i^{-1} \ -1], \quad (18)$$

where $B = \mathbf{h}_{i+N+1} \mathbf{h}_{i+N+1}^T - \mathbf{C}^T \mathbf{A}_i^{-1} \mathbf{C}$, $\mathbf{C} = \mathbf{H}_i \mathbf{h}_{i+N+1}^T$. So, $\mathbf{A}_{i+1}^{-1}$ can be calculated based on $\mathbf{A}_i^{-1}$, which reduces computational cost largely. Then we have $\mathbf{H}_{i+1}^+$ by substituting (18) into (16).

3.4. Calculate the LOO Error. Let $\mathbf{H}_{i+1}^+$ set up the online LOO model; then the LOO error in i th LOO iteration can be expressed as follows:

$$r_j = t_j - f_j(\mathbf{x}_j) = \frac{t_j - \mathbf{H} \mathbf{x}_j \mathbf{H}_{i+1}^{+T}}{1 - (\mathbf{H} \mathbf{x}_j \mathbf{H}_{i+1}^+)_j}, \quad (19)$$

$$j = i + 1, i + 2, \dots, i + N + 1.$$

Then we can obtain the corresponding LOO error, r_j , of each sample from (19). According to the value of r_j , where $j = i + 1, i + 2, \dots, i + N + 1$, we set the relevant weight w_j of each sample, where $0.98 \leq w_j \leq 1$. Note that the smaller r_j is, the bigger w_j is. To emphasize the newest sample and make the decision model simple, we reset the weight w_{i+N+1} of the newest sample $(\mathbf{x}_{i+N+1}, t_{i+N+1})$ as w'_{i+N+1} . This paper defines w'_{i+N+1} as 1.02. And we set the weight w_{i+1} of the oldest sample $(\mathbf{x}_{i+1}, t_{i+1})$ as zero; namely, we set its contribution to the model as zero.

3.5. Weighted Training. After adding the new sample $(\mathbf{x}_{i+N+1}, t_{i+N+1})$, we set the weight w_{i+1} of oldest sample $(\mathbf{x}_{i+1}, t_{i+1})$ as zero, namely, excluding this sample. After excluding $(\mathbf{x}_{i+1}, t_{i+1})$, the output vector becomes $\mathbf{T}_{i+2} = [t_{i+2} \cdots t_{i+N+1}]^T$, and the hidden matrix becomes $\mathbf{H}_{i+2} = [\mathbf{h}_{i+2}^T \cdots \mathbf{h}_{i+N+1}^T]^T$. Then we have

$$\mathbf{H}_{i+2}^+ = \mathbf{H}_{i+2}^T (\mathbf{H}_{i+2} \mathbf{H}_{i+2}^T)^{-1}. \quad (20)$$

Let $\mathbf{A}_{i+2} = \mathbf{H}_{i+2} \mathbf{H}_{i+2}^T$; then

$$\mathbf{H}_{i+2}^+ = \mathbf{H}_{i+2}^T \mathbf{A}_{i+2}^{-1}. \quad (21)$$

From (21), $\mathbf{H}_{i+2}^+$ contains two parts: $\mathbf{H}_{i+2}^T$ and $\mathbf{A}_{i+2}^{-1}$. Because the calculation of $\mathbf{A}_{i+2}^{-1}$ involves matrix inversion, we only set weight on $\mathbf{H}_{i+2}^T$ in order to avoid the huge

computational cost in calculating LOO error. Then the hidden matrix

$$\mathbf{H}_{i+2} = [\mathbf{h}_{i+2}; \mathbf{h}_{i+3}; \dots; \mathbf{h}_{i+N}; \mathbf{h}_{i+N+1}] \quad (22)$$

becomes

$$\mathbf{H}_{i+2} = \begin{bmatrix} \mathbf{h}_{i+2} \\ \mathbf{h}_{i+3} \\ \vdots \\ \mathbf{h}_{i+N} \\ \mathbf{h}_{i+N+1} \end{bmatrix} \cdot \begin{bmatrix} w_{i+2} \\ w_{i+3} \\ \vdots \\ w_{i+N} \\ w_{i+N+1} \end{bmatrix} = \begin{bmatrix} w_{i+2}\mathbf{h}_{i+2} \\ w_{i+3}\mathbf{h}_{i+3} \\ \vdots \\ w_{i+N}\mathbf{h}_{i+N} \\ w_{i+N+1}\mathbf{h}_{i+N+1} \end{bmatrix}. \quad (23)$$

And $\mathbf{H}_{i+2}^T$ becomes

$$\begin{aligned} \mathbf{H}_{i+2}^T &= [w_{i+2}\mathbf{h}_{i+2}^T, w_{i+3}\mathbf{h}_{i+3}^T, \dots, w_{i+N}\mathbf{h}_{i+N}^T, w_{i+N+1}\mathbf{h}_{i+N+1}^T], \end{aligned} \quad (24)$$

where $w_{i+2}, w_{i+3}, \dots, w_{i+N}, w_{i+N+1}$ are the corresponding weights $0.98 \leq w_j \leq 1$, $j = i+2, i+3, \dots, i+N$, and $w_{i+N+1} = w'_{i+N+1} = 1.02$, because

$$\begin{aligned} \mathbf{A}_{i+1} &= \mathbf{H}_{i+1}\mathbf{H}_{i+1}^T = [\mathbf{h}_i^T \quad \mathbf{H}_{i+2}^T]^T [\mathbf{h}_i^T \quad \mathbf{H}_{i+2}^T] \\ &= \begin{bmatrix} \mathbf{h}_i\mathbf{h}_i^T & \mathbf{h}_i\mathbf{H}_{i+2}^T \\ \mathbf{H}_{i+2}\mathbf{h}_i^T & \mathbf{H}_{i+2}\mathbf{H}_{i+2}^T \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{h}_i\mathbf{h}_i^T & \mathbf{h}_i\mathbf{H}_{i+2}^T \\ \mathbf{H}_{i+2}\mathbf{h}_i^T & \mathbf{A}_{i+2} \end{bmatrix}. \end{aligned} \quad (25)$$

From (25), there is a relationship between $\mathbf{A}_{i+2}$ and $\mathbf{A}_{i+1}$. So $\mathbf{A}_{i+2}^{-1}$ can be calculated on the basis of $\mathbf{A}_{i+1}^{-1}$ to simplify the calculation.

Assume that $\mathbf{A}_{i+1}^{-1}$ can be partitioned and expressed as follows:

$$\mathbf{A}_{i+1}^{-1} = \begin{bmatrix} a & \mathbf{F}^T \\ \mathbf{F} & \mathbf{G} \end{bmatrix}, \quad (26)$$

where $a \in \mathbf{R}$, $\mathbf{F} \in \mathbf{R}^N$, and $\mathbf{G} \in \mathbf{R}^{N \times N}$.

As in (25), let $\mathbf{g} = \mathbf{h}_i\mathbf{h}_i^T$, $\mathbf{P} = \mathbf{h}_i\mathbf{H}_{i+2}^T$; then (25) is equivalent to

$$\mathbf{A}_{i+1} = \begin{bmatrix} \mathbf{g} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{A}_{i+2} \end{bmatrix}. \quad (27)$$

By the definition of matrix inversion, $\mathbf{A}_{i+1}\mathbf{A}_{i+1}^{-1} = \mathbf{E}_{(N+1) \times (N+1)}$; namely,

$$\begin{bmatrix} \mathbf{g} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{A}_{i+2} \end{bmatrix} \begin{bmatrix} a & \mathbf{F}^T \\ \mathbf{F} & \mathbf{G} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_{N \times N} \end{bmatrix}. \quad (28)$$

Through the block matrix multiplication, we have

$$\begin{aligned} \mathbf{P}^T a + \mathbf{A}_{i+2} \mathbf{F} &= \mathbf{0}, \\ \mathbf{P}^T \mathbf{F}^T + \mathbf{A}_{i+2} \mathbf{G} &= \mathbf{E}_{N \times N}. \end{aligned} \quad (29)$$

Calculating (29), we have

$$\mathbf{A}_{i+2}^{-1} = \mathbf{G} - \frac{\mathbf{F}\mathbf{F}^T}{a}. \quad (30)$$

Thus, we have $\mathbf{H}_{i+2}^+$ by substituting (24) and (30) into (21).

Then we can update the network weights according to the following equation:

$$\boldsymbol{\beta} = \mathbf{H}_{i+2}^+ \mathbf{T}_{i+2}. \quad (31)$$

3.6. Algorithm. The algorithm of LW-OSELM can be described into two steps.

Step 1 (The initial stage of training). (1) For the initial training sample set $D_0 = \{(\mathbf{x}_i, t_i) \mid i = 1, 2, \dots, N_0\}$, calculate the network weights based on the N_0 training samples by the following equation:

$$\begin{aligned} \boldsymbol{\beta}_0 &= \mathbf{H}_0^+ \mathbf{T}_0, \\ \mathbf{H}_0^+ &= \mathbf{H}_0^T \mathbf{A}_0^{-1}, \\ \mathbf{A}_0 &= \mathbf{H}_0 \mathbf{H}_0^T, \end{aligned} \quad (32)$$

where $\mathbf{H}_0$ is the hidden layer matrix based on new training set and $\mathbf{T}_0 = [t_1, t_2, \dots, t_{N_0}]$.

(2) Set $k = 0$.

Step 2 (Online learning stage). (1) Set the $(k+1)$ th arrived sample as $(\mathbf{x}_{N_0+k+1}, t_{N_0+k+1})$. Now the output vector becomes

$\mathbf{T}_{k+1} = [\mathbf{T}_k^T \quad t_{N_0+k+1}]^T$, and the hidden layer matrix becomes $\mathbf{H}_{k+1} = [\mathbf{H}_k^T \quad \mathbf{h}_{N_0+k+1}^T]^T$. Calculate $\mathbf{H}_{k+1}^+$:

$$\mathbf{H}_{k+1}^+ = \mathbf{H}_{k+1}^T \mathbf{A}_{k+1}^{-1}, \quad (33)$$

where

$$\mathbf{A}_{k+1}^{-1} = \begin{bmatrix} \mathbf{A}_k^{-1} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \frac{1}{B} \begin{bmatrix} \mathbf{A}_k^{-1} \mathbf{C} \\ -1 \end{bmatrix} [\mathbf{C}^T \mathbf{A}_k^{-1} \quad -1],$$

$$\mathbf{B} = \mathbf{h}_{N_0+k+1} \mathbf{h}_{N_0+k+1}^T - \mathbf{C}^T \mathbf{A}_k^{-1} \mathbf{C}, \quad (34)$$

$$\mathbf{C} = \mathbf{H}_k \mathbf{h}_{N_0+k+1}^T.$$

(2) Calculate the LOO error of each sample by the following equation:

$$r_j = t_j - f_j(\mathbf{x}_j) = \frac{t_j - \mathbf{H}\mathbf{x}_j \mathbf{H}_{k+1}^+ \mathbf{T}}{1 - (\mathbf{H}\mathbf{x}_j \mathbf{H}_{k+1}^+)_j}, \quad (35)$$

$$j = k+1, k+2, \dots, N_0+k+1.$$

Then each sample is setting the corresponding weight w_j according to the value of r_j , where $0.98 \leq w_j \leq 1$, and the smaller the r_j is, the bigger the w_j is. In similar way, let $w_{k+1} = 0$; $w_{N_0+k+1} = w'_{i+N+1} = 1.02$.

(3) Set the oldest samples weight $w_{k+1} = 0$. Now the output vector becomes $\mathbf{T}_{k+2} = [t_{k+2} \cdots t_{N_0+k+1}]^T$, and the hidden layer matrix becomes

$$\mathbf{H}_{k+2} = [w_{k+2}\mathbf{h}_{k+2}; w_{k+3}\mathbf{h}_{k+3}; \dots, w_{N_0+k}\mathbf{h}_{N_0+k}; w_{N_0+k+1}\mathbf{h}_{N_0+k+1}]. \quad (36)$$

Calculate $\mathbf{H}_{k+2}^+$ by the following equation:

$$\mathbf{H}_{k+2}^+ = \mathbf{H}_{k+2}^T \mathbf{A}_{k+2}^{-1}. \quad (37)$$

Make $\mathbf{A}_{k+1}^{-1}$ represented as a new block:

$$\mathbf{A}_{k+1}^{-1} = \begin{bmatrix} a & \mathbf{F}^T \\ \mathbf{F} & \mathbf{G} \end{bmatrix}. \quad (38)$$

In (37), consider

$$\begin{aligned} \mathbf{A}_{k+2}^{-1} &= \mathbf{G} - \frac{\mathbf{F}\mathbf{F}^T}{a}, \\ \mathbf{H}_{k+2}^T &= [w_{k+2}\mathbf{h}_{k+2}^T, w_{k+3}\mathbf{h}_{k+3}^T, \dots, w_{N_0+k}\mathbf{h}_{N_0+k}^T, \\ &\quad w_{N_0+k+1}\mathbf{h}_{N_0+k+1}^T]. \end{aligned} \quad (39)$$

Then substitute (39) into (37) to get $\mathbf{H}_{k+2}^+$.

(4) Use (40) to update the network weight:

$$\beta_{k+1} = \mathbf{H}_{k+2}^+ \mathbf{T}_{k+2}. \quad (40)$$

(5) Let $\mathbf{H}_k = \mathbf{H}_{k+2}$, $\mathbf{T}_k = \mathbf{T}_{k+2}$, and $\mathbf{A}_k^{-1} = \mathbf{A}_{k+2}^{-1}$. Let $k = k + 1$, and go to step (2).

For better understanding, here we provide a flow chart of the proposed algorithm, shown as in Figure 1.

4. Experimental Results

In this section, we run experiments to test the proposed algorithm. Our goal is to demonstrate that the proposed algorithm can efficiently improve the generalization performance of OS-ELM in time series prediction. For comparison, we choose two baselines: the classical ELM [9] and OS-ELM [12]. The proposed OS-ELM algorithm with fast LOO weighted strategy is named as LW-OSELM.

For completeness, we examine two types of nonstationary time series data sets. We start with two benchmark chaotic time series data. We further present results on a real-world data set, that is, air pollutants forecasting in Macau [27]. These data sets are described below. The goal is to test the generalization performance on time series data as well as the running speed. In each experiment, all results are the mean of 100 trials. RBF activation function is used in each algorithm. Each variable is linearly rescaled. A method with higher classification accuracy is better.

FIGURE 1: Flow chart of the proposed algorithm.

4.1. Chaotic Time Series. To verify the effectiveness of LW-OSELM for online time series prediction, we take the simulation experiment on three benchmark chaotic time series data: Mackey-Glass, Henon, and Lorenz.

As a chaotic map widely used, Mackey-Glass time series is shown as the following equation:

$$\frac{dx(t)}{dt} = \frac{ax(t - \text{TAU})}{1 + x^{10}(t - \text{TAU})} - bx(t). \quad (41)$$

The initial values are $a = 0.2$, $b = 0.1$, and $\text{TAU} = 30$. Here we generate a set of 1300 points with the first 700 points for

training and the last 600 points for test. τ is set as 1, and the embedding dimension m is set as 4.

Henon time series is shown as the following equation (42):

$$\begin{aligned} x(k+1) &= 1 - A(x(k))^2 + y(k), \\ y(k+1) &= Bx(k). \end{aligned} \quad (42)$$

The initial values are $A = 1.4$, $B = 0.3$, $x(0) = 0.03$, and $y(0) = 0.02$. We generate a set of 600 points with the first 500 points for training and last 100 points for test. In phase reconstruction stage, τ is set as 1, and the embedding dimension m is set as 3.

Lorenz time series is shown as the following equation:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= rx - y - xz, \\ \frac{dz}{dt} &= -bz + xy. \end{aligned} \quad (43)$$

The initial values are $\sigma = 16$, $b = 4$, $r = 45.92$, $x(0) = 0.03$, and $y(0) = 0.02$. We generate a set of 1600 points with the first 600 points for training and last 1000 points for test. In phase reconstruction stage, τ is set as 1, and the embedding dimension m is set as 6.

First, we report the numerical results on the three standard data sets. The mean RMSE of 100 trials on the three data sets are listed in Tables 1, 2, and 3. Here the numbers of neurons are 16, 25, and 25, respectively.

As shown in Tables 1 to 3, although the training time of LW-OSELM is the longest, it gets not only the least training errors, but also the least test errors compared to the others, which is the key factor to measure the performance of a model. Specifically speaking, on Mackey-Glass data, LW-OSELM gets 40% improvement against ELM and 10.6% against OS-ELM in terms of test error. These two values are 4.8% and 4.4%, respectively. In addition, the training times of LW-OSELM on two data sets are 2.844 and 0.2194, respectively, which is in an acceptable range, so it does not affect the model's performance too much. Therefore, LW-OSELM has better performance than the ELM and OS-ELM.

As the only adjustable parameters, it is necessary to test the performance with the change of hidden neurons. Figures 2 and 3, respectively, illustrate the change of the training error and test error of LW-OSELM, OS-ELM, and ELM, with different number of hidden neurons.

From Figures 2 and 3, it is obvious that the change trends of training error and test error of three algorithms are roughly the same, which shows that LW-OSELM is meaningful, and LW-OSELM has smaller errors than the others with the same hidden neurons in most cases. So we can declare that the LW-OSELM has better performance.

In order to further prove the comparative results, we illustrate the predictive performance of three models on data set Mackey-Glass, Henon, and Lorenz, as shown in Figures 4, 5, and 6, respectively.

TABLE 1: Result of three algorithms on Mackey-Glass data set.

	ELM	OS-ELM	LW-OSELM
Training time(s)	0.0052	0.0832	2.8444
Test time(s)	0.0009	0.0021	0.0156
Training error	0.0072	0.0046	0.0041
Test error	0.0070	0.0047	0.0042

TABLE 2: Result of three algorithms on Henon data set.

	ELM	OS-ELM	LW-OSELM
Training time(s)	0.0070	0.0507	0.0554
Test time(s)	0.0062	0.0031	0.0023
Training error	0.0040	0.0050	0.0033
Test error	0.0042	0.0052	0.0034

TABLE 3: Result of three algorithms on Lorenz data set.

	ELM	OS-ELM	LW-OSELM
Training time(s)	0.1856	0.0585	5.6527
Test Time(s)	0.0273	0.0250	0.0296
Training error	0.0022	0.0012	0.0013
Test error	0.0030	0.0020	0.0019

Note that the figures on the right column enlarge the predictive results for more illustration. The original data have no noise, so the predictive curve of three models are all very close to the real curve. From Figures 4(a) to 6(a), it is obvious that the predictive results of three models are roughly the same with real values, which proves the rationality of ELM-based methods. Furthermore, in enlarged part (we take it randomly) from Figures 4(b) to 6(b), we can easily find the prediction curve of LW-OSELM is nearer to the real curve than those of ELM and OS-ELM. These experimental results demonstrate that the proposed algorithm has higher generalization ability.

In brief, LW-OSELM has smaller training errors and test errors than ELM and OS-ELM. By calculating the LOO error of each sample, it is feasible to determine their importance. Through setting the corresponding weight for each sample by means of its LOO error estimation, LW-OSELM is endowed better generalization and stability.

4.2. Macao Meteorological Time Series. In application of air quality monitoring where data often reach in online sequential manner, it is a typical kind of nonstationary time series. In order to further test the stability and generalization of LW-OSELM, we choose suspended particulate matters (PM₁₀) to conduct experiment. Due to the limitation of the acquisition data, this paper adopts the air quality data of Macao meteorological bureau to conduct simulation experiments [24].

FIGURE 2: Training error of three algorithms with different number of hidden neurons for data sets (a) Mackey-Glass, (b) Henon, and (c) Lorenz.

4.2.1. Data Pre-Processing. We define the training data set $D = (x, t)$, where x represents the input variables and t represents the output variables. The input sample is constructed by the suspended particulate matters (PM_{10}), nitrogen dioxide (NO_2), and sulfur dioxide (SO_2); that is, $x = (d(PM_{10}), d(SO_2), d(NO_2), d(O_3))$. And the output sample is the value of the next day's (PM_{10}); that is, $t = d + 1(PM_{10})$. The training data and test data were normalized to a range of $[-1, 1]$, as follows:

$$v' = \frac{2(v - v_{\min})}{(v_{\max} - v_{\min})} - 1, \quad (44)$$

where v is a variable in x or t . $v_{\max}$ and $v_{\min}$ are the minimum value and maximum value of v in study period, respectively.

4.2.2. Experiment Result. To verify the validity of the LW-OSELM for time series prediction, we use the data collected from 2010 to 2013 to conduct experiment. Specifically, the data in 2010 are used for initial offline training, the data in 2011 are used for online training, the data in 2012 are used for test, and the data in 2013 are used for testing the generalization performance.

Considering the deficiency of traditional OS-ELM in time series prediction, we put forward a new kind of weighted extreme learning machine based on online leave-one-out

FIGURE 3: Test error of three algorithms with different number of hidden neurons for data sets (a) Mackey-Glass, (b) Henon, and (c) Lorenz.

cross-validation. According to the value of the online LOO error estimation, weigh the corresponding weight for each sample. To verify the rationality of this dynamic weight-setting strategy, it is necessary to compare LW-OSELM and the fixed-weighting OS-ELM (namely, WELM). The WELM will set weight value as 0.98 for all old samples and 1.02 for the latest sample. We set hidden neurons as 25. The comparison of LW-OSELM and WELM is illustrated in Table 4.

From Table 4, compared with WELM, the training error and test error of LW-OSELM are much smaller, which demonstrates that the dynamic weight-setting strategy, that is, setting weights on samples according to their online LOO errors, is feasible.

TABLE 4: Comparative results of LW-OSELM and WELM.

	LW-OSELM	WELM
Training time(s)	2.9703	0.4056
Test time(s)	0.0094	0
Training error	0.1767	0.4276
Test error	0.1794	0.4310

We compare the performance of ELM, OS-ELM, and LW-OSELM. Given the hidden layer activation function of RBF kernel function, the mean RMES of 100 trials on

FIGURE 4: Comparative results of three models on data set Mackey-Glass (a) original figure and (b) enlarged figure.

FIGURE 5: The comparison of real value and test values of three models for data set Henon (a) original figure and (b) enlarged figure.

Macao meteorological time series is listed in Table 5. Here the numbers of neurons are 15.

By contrast, LW-OSELM has little smaller training error and test error while its training time is a little longer. Note that, in this experiment, the comparative results are not remarkable like in Section 4.1. The reason is quite likely that we did not employ embedding dimension, that is, reconstructing phase space like in (1). Here we merely use the data in the latest day as input sample, rather than using the data of past few days.

We also examine the effect of hidden neurons. Figures 7 and 8, respectively, show the change of training error and test error with different number of hidden neurons.

TABLE 5: Comparative results of three models.

	ELM	OS-ELM	LW-OSELM
Training time(s)	0.0125	0.0998	3.3961
Test time(s)	0.0031	0.0094	0.0012
Training error	0.1706	0.1721	0.1705
Test error	0.1811	0.1808	0.1739

From Figures 7 and 8, the training error and test error of LW-OSELM are both smaller than the others with the most hidden neurons. ELM tends to be the most unstable

FIGURE 6: The comparison of real value and test values of three models for data set Lorenz (a) original figure and (b) enlarged figure.

FIGURE 7: Training error of three models with different number of hidden neurons.

FIGURE 8: Test error of three models with different number of hidden neurons.

with drastic fluctuation. And LW-OSELM performs similarly stable to OS-ELM, which keeps pace with the results in Table 5.

Moreover, we report the generalization performance of three algorithms with different prediction step, as in Figure 9.

Obviously, LW-OSELM is better than the others. For further clarification, we also compare the mean and variance of three models. The mean of ELM, OS-ELM, and LW-OSELM is, respectively, 0.0726, 0.0617, and 0.0605, and the variance is, respectively, 3.8340×10^{-4} , 3.5584×10^{-4} , and 3.0809×10^{-4} . Obviously, the average error and variance of LW-OSELM are both the least, which indicates that LW-OSELM has better accuracy and stronger stability.

5. Conclusion and Future Work

In this paper, nonstationary time series prediction is addressed. The key idea is distinguishing the importance of samples in time series using its LOO cross-validation error. This idea is a new attempt for weight-setting strategy. To realize this strategy, this paper utilizes OS-ELM as baseline algorithm and proposes a new LOO error estimation which is fast and quite applied to time series prediction. Based on this estimation, this paper proposes a dynamic weight-setting algorithm for OS-ELM. The experimental results on two benchmark chaotic time series data sets and a real-world data set demonstrate the effectiveness of the proposed approach. It is

FIGURE 9: Accuracy of three algorithms with different prediction step.

a matter of choosing more efficient method for calculating leave-one-out error. Another problem is how to extend the proposed algorithm to multi-input multioutput regression, which should be achieved by introducing the corresponding background knowledge and will be studied in our future research.

Conflict of Interests

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

Acknowledgments

The authors wish to thank the author C. M. Vong of [27] for useful discussion and instruction. This work was supported by the National Natural Science Foundation of China (nos. U1204609, 61173071), China Postdoctoral Science Foundation (no. 2014M550508), and Postgraduate Technology Innovation Project of Henan Normal University (no. YL201420).

References

- [1] J. G. de Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International Journal of Forecasting*, vol. 22, no. 3, pp. 443–473, 2006.
- [2] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence*, Warwick, vol. 1981, pp. 366–381, 1980.
- [3] S. Mukherjee, E. Osuna, and F. Girosi, "Nonlinear prediction of chaotic time series using support vector machines," in *Proceedings of the 7th IEEE Workshop on Neural Networks for Signal Processing (NNSP '97)*, pp. 511–520, IEEE Press, Amelia Island, Fla, USA, September 1997.
- [4] K. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1-2, pp. 307–319, 2003.
- [5] D. Du, X. Li, M. Fei, and G. W. Irwin, "A novel locally regularized automatic construction method for RBF neural models," *Neurocomputing*, vol. 98, pp. 4–11, 2012.
- [6] P. Yee and S. Haykin, "A dynamic regularized radial basis function network for nonlinear, nonstationary time series prediction," *IEEE Transactions on Signal Processing*, vol. 47, no. 9, pp. 2503–2521, 1999.
- [7] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, pp. 1163–1168, Hangzhou, China, 2014.
- [8] F. Corona and A. Lendasse, "Variable scaling for time series prediction," in *Proceedings of European Symposium on Time Series Prediction (ESTSP '07)*, pp. 69–76, Espoo, Finland, 2007.
- [9] 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.
- [10] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [11] J. W. Cao, Z. Lin, G. B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, pp. 66–77, 2012.
- [12] 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.
- [13] C.-F. Lin and S.-D. Wang, "Fuzzy support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 464–471, 2002.
- [14] F. E. H. Tay and L. J. Cao, "Modified support vector machines in financial time series forecasting," *Neurocomputing*, vol. 48, pp. 847–861, 2002.
- [15] W. Mao, G. Yan, and L. Dong, "Weighted solution path algorithm of support vector regression based on heuristic weight-setting optimization," *Neurocomputing*, vol. 73, no. 1-3, pp. 495–505, 2009.
- [16] Y. Bao, T. Xiong, and Z. Hu, "Multi-step-ahead time series prediction using multiple-output support vector regression," *Neurocomputing*, vol. 129, pp. 482–493, 2014.
- [17] X. Wang and M. Han, "Online sequential extreme learning machine with kernels for nonstationary time series prediction," *Neurocomputing*, vol. 145, pp. 90–97, 2014.
- [18] A. Grigorievskiy, Y. Miche, A.-M. Ventelä, E. Séverin, and A. Lendasse, "Long-term time series prediction using OP-ELM," *Neural Networks*, vol. 51, pp. 50–56, 2014.
- [19] H. Rong, G.-B. Huang, N. Sundarajan et al., "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 4, pp. 1067–1072, 2009.
- [20] D. M. Allen, "The relationship between variable selection and data augmentation and a method for prediction," *Technometrics*, vol. 16, pp. 125–127, 1974.
- [21] Y. Miche, M. van Heeswijk, P. Bas, O. Simula, and A. Lendasse, "TROP-ELM: a double-regularized ELM using LARS and Tikhonov regularization," *Neurocomputing*, vol. 74, no. 16, pp. 2413–2421, 2011.

- [22] G.-B. Huang and H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 224–229, 1998.
- [23] Y. Lan, Y. C. Soh, and G. B. Huang, "Two-stage extreme learning machine for regression," *Neurocomputing*, vol. 73, no. 16-18, pp. 3028–3038, 2010.
- [24] G. Feng, G.-B. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *IEEE Transactions on Neural Networks*, vol. 20, no. 8, pp. 1352–1357, 2009.
- [25] X.-Y. Liu, P. Li, and C.-H. Gao, "Fast leave-one-out cross-validation algorithm for extreme learning machine," *Journal of Shanghai Jiaotong University*, vol. 45, no. 8, pp. 6–11, 2011.
- [26] X. Zhang and H.-L. Wang, "Local extreme learning machine and its application to condition on-line monitoring," *Journal of Shanghai Jiaotong University*, vol. 45, no. 2, pp. 236–240, 2011.
- [27] C.-M. Vong, W.-F. Ip, P.-K. Wong, and C.-C. Chiu, "Predicting minority class for suspended particulate matters level by extreme learning machine," *Neurocomputing*, vol. 128, pp. 136–144, 2014.