applied sciences

MDPI

Article

Time Series Prediction Based on Adaptive Weight Online Sequential Extreme Learning Machine

Junjie Lu, Jinquan Huang * and Feng Lu

Jiangsu Province Key Laboratory of Aerospace Power Systems, College of Energy and Power Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China; junjielulu@outlook.com (J.L.); lfaann@nuaa.edu.cn (F.L.)

* Correspondence: jhuang@nuaa.edu.cn; Tel.: +86-2584-8959-95

Academic Editor: Chien-Hung Liu

Received: 5 January 2017; Accepted: 20 February 2017; Published: 2 March 2017

Abstract: A novel adaptive weight online sequential extreme learning machine (AWOS-ELM) is proposed for predicting time series problems based on an online sequential extreme learning machine (OS-ELM) in this paper. In real-world online applications, the sequentially coming data chunk usually possesses varying confidence coefficients, and the data chunk with a low confidence coefficient tends to mislead the subsequent training process. The proposed AWOS-ELM can improve the training process by accessing the confidence coefficient adaptively and determining the training weight accordingly. Experiments on six time series prediction data sets have verified that the AWOS-ELM algorithm performs better in generalization performance, stability, and prediction ability than the OS-ELM algorithm. In addition, a real-world mechanical system identification problem is considered to test the feasibility and efficacy of the AWOS-ELM algorithm.

Keywords: time series prediction; extreme learning machine; adaptive weight; online learning

1. Introduction

Time series prediction technology has already been studied over the past few decades, and a large amount of applications have been reported in a wide range of fields, such as weather forecasting [1], stock market prediction [2], communication signal processing [3], sales forecasting [4], and so on. On account of the frequent applications of time series predictions, plenty of predicting methods have been developed. Gooijer and Hyndman gave an overview of various predicting methods and indicated the future directions for time series prediction problems [5]. In particular, the classical statistical linear method, an autoregressive integrated moving average model (ARIMA) based modeling method, is still widely adopted and the complete methodology is able to be found in Box's and Jenkins's remarkable contribution [6]. However, the predicting accuracy of classical statistical methods suffer from the nonlinearity and complexity of many real time series. For this reason, some computational intelligence methods which may outperform classical statistical methods in many complex nonlinear problems have emerged [7,8].

Artificial neural network (ANN) methods have attracted extensive attention in the time series prediction field [9]. ANNs are universal nonlinear regression techniques, and are able to be applied into time series prediction conveniently [10]. Furthermore, compared with classical methods, the assumptions for prediction can be relaxed by ANNs, such as Gaussian distribution of noise and the linearity of the model. In contrast to the fact that the model hyperparameters of the ARIMA model is required to be fine-tuned for good prediction, this complication is able to be avoided by ANNs [11]. The hyperparameters of the ARIMA model are quite often adjusted according to domain knowledge, while the ANNs model usually can obtain competitive results without any domain knowledge [12].

For some situations where intensive human adjustment is not affordable, ANNs and other computational intelligence methods are usually better than classical statistical methods. Kumar investigated neural network models for forecasting the Indian stock market index [13]. Yoon discussed integrating ANN and support vector machine (SVM) methods for long-term prediction, and the stability and accuracy are improved [14]. Nevertheless, since the neural networks used by Kumar and Yoon are obtained in an iterative way by means of gradient-based algorithms, the diagnosis systems suffer from time-consuming problems.

Extreme learning machine (ELM) is a high-efficiency learning algorithm for single-hidden layer feedforward neural network (SLFN), and it has been proven to have classification capacity and universal approximation capacity [15]. In addition, Huang has shown that the hidden-layer parameters can be randomly assigned, and then the output weight is able to be computed analytically [16]. It has been verified that ELM costs much less training time and has better or similar generalization performance than SVM and traditional neural networks [17]. Hosseinioun presented the use of wavelet transform and adaptive ELM to forecast outlier occurrence in stock market time series [18]. Dash presented an optimized ELM for predicting financial time series [19]. The ELM based prediction methods have high accuracy and fast learning speed in off-line cases, but they are not suitable for online applications. Liang et al. proposed OS-ELM by incorporating a sequential learning algorithm with ELM [20]. Compared with conventional online training methods, OS-ELM tends to have a faster training speed and better generalization performance. However, in lots of real-world online applications, the confidence coefficient of a sequential data chunk may be disturbed by measurement noise and external disturbance. If the data chunk with low confidence coefficient is employed in the learning process in a normal way, the accuracy of the trained network is likely to be reduced. In this paper, the AWOS-ELM algorithm is proposed to reduce the negative influence of data chunks with low confidence coefficients, where the confidence coefficient of each data chunk is assessed before being used to train the network, and the weight of each data chunk is obtained according to the assessed confidence coefficient. Experiments on six time series prediction problems and a mechanical system identification problem have verified that the AWOS-ELM algorithm performs better in generalization performance, stability, and predictability than the OS-ELM algorithm.

This manuscript is organized as follows. In Section 2, the basic concepts and related works of optimization ELM and OS-ELM algorithms are reviewed briefly. The integrated structure of the proposed algorithm and the formula derivation is given in Section 3. In Section 4, the performance evaluation of the AWOS-ELM is carried out on six time series prediction problems and a mechanical system identification problem. The conclusion is drawn in Section 5.

2. Preliminaries

In this section, with the purpose of offering preliminaries pertinent to the proposed AWOS-ELM algorithm, the optimization of ELM and OS-ELM is reviewed briefly. OS-ELM, an online learning algorithm on the basis of classical ELM, was proposed by Liang in 2006 for training sequential data. Huang further developed the classical ELM into the optimization ELM according to the Karush-Kuhn-Tucker (KKT) theory and optimization theory in 2012 [21]. Compared to the classical ELM, the regularization parameters are used in the optimization ELM to increase the accuracy and generation performance. The OS-ELM in this paper also employs the regularization parameters according to Huang's theory.

2.1. Optimization Extreme Learning Machine

ELM is a high-efficiency SLFN learning algorithm, where the hidden node parameters can be assigned randomly. Assume that there are N different training data $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N} \subset \Re^n \times \Re^m$ for the

supervised learning process, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \Re^n$ and $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in \Re^m$ are the input vector and output vector, respectively, and the mathematical model of SLFNs is described as:

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i g(\boldsymbol{\omega}_i, b_i, \mathbf{x}), \mathbf{x} \in \Re^n \tag{1}$$

where $\boldsymbol{\omega}_i \in \Re^n$, $b_i \in \Re$ and $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T \in \Re^m$ denote the ith hidden node parameters, L is the hidden nodes number, and $g(\boldsymbol{\omega}_i, b_i, \mathbf{x})$ represents the hidden-layer output in accordance with the input $\mathbf{x}$. If the N training samples are absolutely approximated by the SLFNs with L hidden nodes, it indicates the following equation:

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i g(\boldsymbol{\omega}_i, b_i, \mathbf{x}) = \mathbf{t}_j, j = 1, 2, \ldots, N \tag{2}$$

Equation (2) is able to be described compactly as the following equation:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{3}$$

where

$$\mathbf{H}(\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_L, b_1, \ldots, b_L, \mathbf{x}_1, \ldots, \mathbf{x}_N) = \begin{bmatrix} g(\boldsymbol{\omega}_1, b_1, \mathbf{x}_1) & \cdots & g(\boldsymbol{\omega}_L, b_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ g(\boldsymbol{\omega}_1, b_1, \mathbf{x}_N) & \cdots & g(\boldsymbol{\omega}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N \times L} \tag{4}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_L^T \end{bmatrix}_{L \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{5}$$

Traditionally, for the purpose of training an SLFN, one needs to find specific $\boldsymbol{\omega}_i, b_i, \boldsymbol{\beta}_i, i = 1, \ldots, L$, such that $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|$ takes a minimum value. If $\mathbf{H}$ is unknown, the gradient-based approaches are usually employed to iteratively adjust $\boldsymbol{\omega}_i, b_i, \boldsymbol{\beta}_i$. However, for most applications, the gradient-based method is extremely time-consuming and often stop at the local minimum. According to the theory of Huang, the hidden-layer learning parameters $\boldsymbol{\omega}_i$ and b_i can be assigned randomly, and as such the SLFN is able to approximate any target function universally as soon as the activation function is nonzero, the target function is continuous and the input sets are compact [17]. If $L \leq N$, the column rank of $\mathbf{H}$ is full with probability one, and in real-world applications, the condition $L \leq N$ can be easily satisfied. Considering the norm of the output weight $\boldsymbol{\beta}$ to be part of the cost function [22], the optimization ELM model can be represented as:

$$\begin{aligned} \min : L_{elm} &= \tfrac{1}{2}\|\boldsymbol{\beta}\|^2 + \tfrac{C}{2}\sum_{i=1}^{N} \|\boldsymbol{\varepsilon}_i\|^2 \\ st : \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} &= \mathbf{t}_i - \boldsymbol{\varepsilon}_i, \;\; i = 1, 2, \ldots, N \end{aligned} \tag{6}$$

where $\boldsymbol{\varepsilon}_i$ is the prediction error of the ith training sample. Because the convex optimization problem does not have an inequality constraint, the Slater's condition is satisfied and the strong duality holds. Consequently, on the basis of the KKT theorem, Equation (6) can be represented as follows:

$$\min : \; L_{elm} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{C}{2}\sum_{i=1}^{N} \boldsymbol{\varepsilon}_i^2 - \sum_{i=1}^{N} \boldsymbol{\alpha}_i(\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - \mathbf{t}_i + \boldsymbol{\varepsilon}_i) \tag{7}$$

where $\boldsymbol{\alpha}_i$ is the Lagrange multiplier. By optimizing Equation (7), the output weight can be obtained as follows:

$$\boldsymbol{\beta} = \left(\frac{I}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T} \tag{8}$$

Since the output weights $\boldsymbol{\beta}$ is computed analytically, compared with traditional iterative implementations of SLFNs, the optimization ELM has similar generalization performance and dramatically increased running speed.

2.2. Online Sequential Extreme Learning Machine

In many practical instances, the sequential training samples $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i)|\mathbf{x}_i \in \Re^n, \mathbf{t}_i \in \Re^m, i = 1, 2, \ldots\}$ are produced chunk by chunk, and the chunk size may be fixed or various. Assume that the jth data chunk has N_j samples, then the chunk at time k is able to be represented as $\aleph_k = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=(\sum_{j=0}^{k-1} N_j)+1}^{\sum_{j=0}^{k} N_j}$. The initialization of the learning process is carried out according to a small data chunk $\aleph_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$, where N_0 is the samples number of data chunk $\aleph_0$, and N_0 ought to be equal to or greater than L. With The hidden-layer parameters $(\boldsymbol{\omega}_i, b_i), i = 1, 2, \ldots, L$ assigned into random values, the initial $\mathbf{H}_0$ can be computed as the following equation:

$$\mathbf{H}_0 := \mathbf{H}(\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_L, b_1, \ldots, b_L, \mathbf{x}_1, \ldots, \mathbf{x}_{N_0}) \tag{9}$$

and then the initial $\boldsymbol{\beta}_0$ is able to be obtained according to ELM as follows:

$$\boldsymbol{\beta}_0 = \mathbf{P_0}\mathbf{H}_0^T\mathbf{T}_0 \tag{10}$$

where $\mathbf{P_0} = \left(\frac{I}{C} + \mathbf{H}_0^T\mathbf{H}_0\right)^{-1}$ and $\mathbf{T_0} = \left[\mathbf{t}_1, \ldots, \mathbf{t}_{N_0}\right]^T$.

The partial hidden-layer output matrixes $\mathbf{h}_{k+1}$ and the partial output-layer matrixes $\mathbf{t}_{k+1}$ corresponding to data chunk at time $k + 1$ are respectively defined as:

$$\mathbf{h}_{k+1} := \mathbf{H}\left(\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_L, b_1, \ldots, b_L, \mathbf{x}_{(\sum_{j=0}^{k} N_j)+1}, \ldots, \mathbf{x}_{\sum_{j=0}^{k+1} N_j}\right) \tag{11}$$

$$\mathbf{t}_{k+1} := \left[\mathbf{t}_{(\sum_{j=0}^{k} N_j)+1}, \ldots, \mathbf{t}_{\sum_{j=0}^{k+1} N_j}\right]^T \tag{12}$$

Then $\mathbf{H}_k$ and $\mathbf{T}_k$ can be respectively expressed as:

$$\mathbf{H}_k = \mathbf{H}\left(\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_L, b_1, \ldots, b_L, \mathbf{x}_1, \ldots, \mathbf{x}_{\sum_{j=0}^{k} N_j}\right), \mathbf{T}_k = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_{\sum_{j=0}^{k} N_j}^T \end{bmatrix} \tag{13}$$

and we have

$$\mathbf{H}_{k+1} = \begin{bmatrix} \mathbf{H}_k \\ \mathbf{h}_{k+1} \end{bmatrix}, \mathbf{T}_{k+1} = \begin{bmatrix} \mathbf{T}_k \\ \mathbf{t}_{k+1}^T \end{bmatrix} \tag{14}$$

The least squares solution of $\mathbf{H}_{k+1}\boldsymbol{\beta} = \mathbf{T}_{k+1}$ should be the output weight at time $k + 1$, $\boldsymbol{\beta}_{k+1}$ and it is able to be computed in an iterative way as follows:

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \mathbf{P}_{k+1}\mathbf{h}_{k+1}^T\left(\mathbf{t}_{k+1}^T - \mathbf{h}_{k+1}\boldsymbol{\beta}_k\right) \tag{15}$$

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k\mathbf{h}_{k+1}^T\left(I + \mathbf{h}_{k+1}\mathbf{P}_k\mathbf{h}_{k+1}^T\right)^{-1}\mathbf{h}_{k+1}\mathbf{P}_k \tag{16}$$

OS-ELM is composed of an initialization phase and sequential learning phase and need not retain all the historic data. In the initialization phase, $\mathbf{H}_0$, $\boldsymbol{\beta}_0$, $\mathbf{P_0}$, and $\mathbf{T_0}$ are initialized for the use in the sequential learning phase. The samples number of the initialization chunk should be equal to or greater than the hidden nodes number. In the sequential learning phase, the sequential date chunk is commenced on iteratively. Once the training process on the latest coming data chunk is completed, the historic data can be discarded and not used any more. From the derivation of OS-ELM, it is easy to find that OS-ELM and ELM have similar generalization performances. In fact, the ELM algorithm is a specific example of the OS-ELM algorithm if all of the training samples are processed in the initialization of the learning process.

3. The Proposed Adaptive Weight Online Sequential Extreme Learning Machine

In lots of real online applications, with measurement noise and unexpected external disturbance, the sequential data chunks often have varying confidence coefficients. It can be easily found that OS-ELM cannot deal with the varying confidence coefficients very well. If a data chunk with a low confidence coefficient is employed to train the network in the normal way, the accuracy of the trained network is likely to be reduced. In this section, we propose the novel AWOS-ELM, where the confidence coefficient of each data chunk is assessed before being used to train the network, and the weight of each data chunk is obtained accordingly.

3.1. Integrated Structure

The block diagram of AWOS-ELM algorithm is given in Figure 1. When the new sequential sample $\aleph_{k+1} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=(\sum_{j=0}^{k} N_j)+1}^{\sum_{j=0}^{k+1} N_j}$ arrives, the weight estimator accesses the confidence coefficient of $\aleph_{k+1}$ and determines the corresponding weight λ_{k+1}. Then the training module of AWOS-ELM algorithm utilizes λ_{k+1} and $\aleph_{k+1}$ to train the network. The weight estimator includes an AWOS-ELM testing module, residual generator and sigmoid function. The testing module produces the prediction value $\{\hat{\mathbf{t}}_i\}_{i=(\sum_{j=0}^{k} N_j)+1}^{\sum_{j=0}^{k+1} N_j}$ according to the input vector $\{\mathbf{x}_i\}_{i=(\sum_{j=0}^{k} N_j)+1}^{\sum_{j=0}^{k+1} N_j}$. The residual r_{k+1} is produced by the residual generator according to the comparison between the prediction value $\{\hat{\mathbf{t}}_i\}_{i=(\sum_{j=0}^{k} N_j)+1}^{\sum_{j=0}^{k+1} N_j}$ and the target value $\{\mathbf{t}_i\}_{i=(\sum_{j=0}^{k} N_j)+1}^{\sum_{j=0}^{k+1} N_j}$, where the residual r_{k+1} is defined as the following:

$$r_{k+1} = \sqrt{\frac{\sum_{i=1}^{N_{k+1}} \left\| \hat{\mathbf{t}}_{i+\sum_{j=0}^{k} N_j} - \mathbf{t}_{i+\sum_{j=0}^{k} N_j} \right\|_F^2}{N_{k+1} \times m}} \tag{17}$$

where m is the dimension of the output vector $\mathbf{t}_i$. Then the sigmoid mapper produce the accessed weight λ_{k+1} according to the difference between τ_r and r_{k+1} as the following:

$$\lambda_{k+1} = \frac{1}{1 + e^{-\varphi(\tau_r - r_{k+1})}} \tag{18}$$

where τ_r is the threshold value and φ is a scaling factor which can represent the gradient of the sigmoid function at the threshold point. For the samples with the residuals closer to the threshold τ_r, the greater the φ is, the greater the difference between calculated weights will be. Moreover, φ is manually selected to be 500 according to the performance in this paper. As observed from Equation (18), the greater the residual r_{k+1} is, the less the accessed weight λ_{k+1} is, and $0 < \lambda_k < 1$. The network of the testing module should be updated according to the training module before next sequential data chunk is incoming. The abnormal samples which cannot match the normal model well will have low accessed weight, and their negative impact to the subsequent learning process will be reduced. Thus, the AWOS-ELM algorithm is able to properly handle the varying confidence coefficients of each data chunk.

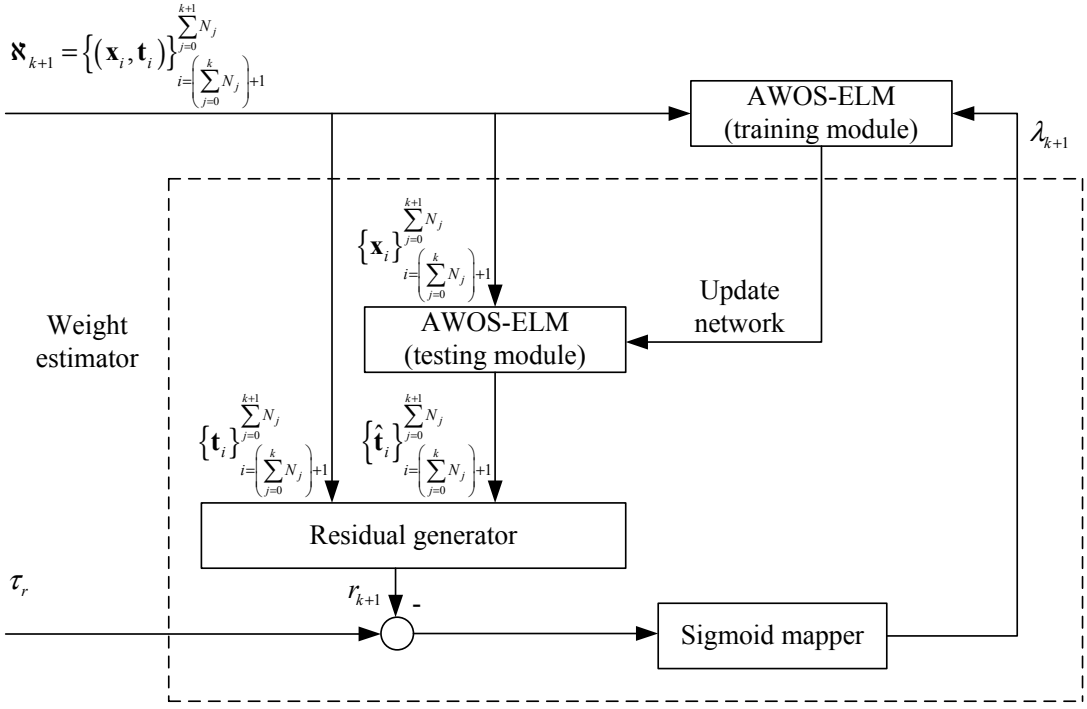

Figure 1. Block diagram of the adaptive weight online sequential extreme learning machine (AWOS-ELM) algorithm.

3.2. Formula Derivation

The assessed weight of the data chunk at time k is λ_k, then $\boldsymbol{\beta}'_{k+1}$ is the least squares solution of the following equation:

$$\begin{bmatrix} \lambda_0\mathbf{H}_0 \\ \lambda_1\mathbf{h}_1 \\ \vdots \\ \lambda_{k+1}\mathbf{h}_{k+1} \end{bmatrix} \boldsymbol{\beta}' = \begin{bmatrix} \lambda_0\mathbf{T}_0 \\ \lambda_1\mathbf{t}_1^T \\ \vdots \\ \lambda_{k+1}\mathbf{t}_{k+1}^T \end{bmatrix} \tag{19}$$

Let $\mathbf{H}'_{k+1} := \begin{bmatrix} \mathbf{H}'_k \\ \lambda_{k+1}\mathbf{h}_{k+1} \end{bmatrix}$, $\mathbf{H}'_0 := \lambda_0\mathbf{H}_0$, $\mathbf{T}'_{k+1} := \begin{bmatrix} \mathbf{T}'_k \\ \lambda_{k+1}\mathbf{t}_{k+1}^T \end{bmatrix}$, $\mathbf{T}'_0 := \lambda_0\mathbf{T}_0$, then Equation (19) can be described in a compact way as:

$$\mathbf{H}'_{k+1}\boldsymbol{\beta}' = \mathbf{T}'_{k+1} \tag{20}$$

Theorem 1. *The solution of Equation (20) in the sense of least squares is able to be obtained in an iterative way as follows:*

$$\boldsymbol{\beta}'_{k+1} = \boldsymbol{\beta}'_k + \mathbf{K}_{k+1}\left(\mathbf{t}_{k+1} - \mathbf{h}_{k+1}\boldsymbol{\beta}'_k\right) \tag{21}$$

$$\mathbf{P}'_{k+1} = (I - \mathbf{K}_{k+1}\mathbf{h}_{k+1})\mathbf{P}'_k \tag{22}$$

$$\mathbf{K}_{k+1} = \mathbf{P}'_k\mathbf{h}_{k+1}\left(\frac{I}{\lambda^2_{k+1}} + \mathbf{h}_{k+1}\mathbf{P}'_k\mathbf{h}^T_{k+1}\right)^{-1} \tag{23}$$

where $\mathbf{P}'_k := \left(\mathbf{H}'^T_k\mathbf{H}'_k\right)^{-1}$, $\boldsymbol{\beta}'_k = \mathbf{P}'_k\mathbf{H}'^T_k\mathbf{T}'_k$.

Proof. According to the definition of $\mathbf{P}'_k$, $\mathbf{P}'_{k+1}$ can be found as follows:

$$\mathbf{P}'_{k+1} = \left(\begin{bmatrix} \mathbf{H}'_k \\ \lambda_{k+1}\mathbf{h}_{k+1} \end{bmatrix}^T \begin{bmatrix} \mathbf{H}'_k \\ \lambda_{k+1}\mathbf{h}_{k+1} \end{bmatrix}\right)^{-1} = \left(\mathbf{H}'^T_k\mathbf{H}'_k + \lambda^2_{k+1}\mathbf{h}^T_{k+1}\mathbf{h}_{k+1}\right)^{-1} \tag{24}$$

Apply the Sherman-Morrison-Woodbury formula [23] into Equation (24), and then $\mathbf{P}'_{k+1}$ can be determined iteratively as follows:

$$\begin{aligned} \mathbf{P}'_{k+1} &= \mathbf{P}'_k - \mathbf{P}'_k\mathbf{h}^T_{k+1}\left(\tfrac{I}{\lambda^2_{k+1}} + \mathbf{h}_{k+1}\mathbf{P}'_k\mathbf{h}^T_{k+1}\right)^{-1}\mathbf{h}_{k+1}\mathbf{P}'_k \\ &= \mathbf{P}'_k - \mathbf{K}_{k+1}\mathbf{h}_{k+1}\mathbf{P}'_k \end{aligned} \tag{25}$$

According to Equation (25) and $\boldsymbol{\beta}'_{k+1} = \mathbf{P}'_{k+1}\mathbf{H}'^T_{k+1}\mathbf{T}'_{k+1}$, the output matrix $\boldsymbol{\beta}'_{k+1}$ can be obtained from the following equation:

$$\begin{aligned} \boldsymbol{\beta}'_{k+1} &= \left(\mathbf{P}'_k - \mathbf{K}_{k+1}\mathbf{h}_{k+1}\mathbf{P}'_k\right)\begin{bmatrix} \mathbf{H}'_k \\ \lambda_{k+1}\mathbf{h}_{k+1} \end{bmatrix}^T \begin{bmatrix} \mathbf{T}'_k \\ \lambda_{k+1}\mathbf{t}^T_{k+1} \end{bmatrix} \\ &= \left(\mathbf{P}'_k - \mathbf{K}_{k+1}\mathbf{h}_{k+1}\mathbf{P}'_k\right)\left(\mathbf{H}'^T_k\mathbf{T}'_k + \lambda^2_{k+1}\mathbf{h}^T_{k+1}\mathbf{t}^T_{k+1}\right) \\ &= \mathbf{P}'_k\mathbf{H}'^T_k\mathbf{T}'_k - \mathbf{K}_{k+1}\mathbf{h}_{k+1}\mathbf{P}'_k\mathbf{H}'^T_k\mathbf{T}' \\ &\quad +\lambda^2_{k+1}\left(\mathbf{P}'_k\mathbf{h}^T_{k+1} - \mathbf{K}_{k+1}\mathbf{h}_{k+1}\mathbf{P}'_k\mathbf{h}^T_{k+1}\right)\mathbf{t}^T_{k+1} \\ &= \boldsymbol{\beta}'_k - \mathbf{K}_{k+1}\mathbf{h}_{k+1}\boldsymbol{\beta}'_k \\ &\quad +\lambda^2_{k+1}\left(\mathbf{P}'_k\mathbf{h}^T_{k+1} - \mathbf{K}_{k+1}\mathbf{h}_{k+1}\mathbf{P}'_k\mathbf{h}^T_{k+1}\right)\mathbf{t}^T_{k+1} \end{aligned} \tag{26}$$

Then we can simplify the $\mathbf{P}'_k\mathbf{h}(k+1)^T - \mathbf{K}_{k+1}\mathbf{h}(k+1)\mathbf{P}'_k\mathbf{h}(k+1)^T$ in Equation (26) as:

$$\begin{aligned} &\mathbf{P}'_k\mathbf{h}^T_{k+1} - \mathbf{K}_{k+1}\mathbf{h}_{k+1}\mathbf{P}'_k\mathbf{h}^T_{k+1} \\ &= \mathbf{P}'_k\mathbf{h}^T_{k+1} - \mathbf{K}_{k+1}\mathbf{h}_{k+1}\mathbf{P}'_k\mathbf{h}^T_{k+1} - \tfrac{\mathbf{K}_{k+1}}{\lambda^2_{k+1}} + \tfrac{\mathbf{K}_{k+1}}{\lambda^2_{k+1}} \\ &= \mathbf{P}'_k\mathbf{h}^T_{k+1} - \mathbf{K}_{k+1}\left(\tfrac{I}{\lambda^2_{k+1}} + \mathbf{h}_{k+1}\mathbf{P}'_k\mathbf{h}^T_{k+1}\right) + \tfrac{\mathbf{K}_{k+1}}{\lambda^2_{k+1}} \\ &= \tfrac{\mathbf{K}_{k+1}}{\lambda^2_{k+1}} \end{aligned} \tag{27}$$

Substituting Equation (27) into Equation (26), $\boldsymbol{\beta}'_{k+1}$ is able to be determined in a compact way as:

$$\boldsymbol{\beta}'_{k+1} = \boldsymbol{\beta}'_k + \mathbf{K}_{k+1}\left(\mathbf{t}^T_{k+1} - \mathbf{h}_{k+1}\boldsymbol{\beta}'_k\right) \tag{28}$$

Proposed AWOS-ELM Algorithm: If L and g:$\Re \to \Re$ are given, the AWOS-ELM algorithm is able to be summarized as follows:

Step 1. Initialization phase: choose the initial data chunk $\aleph_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$, where $N_0 \geq L$.

(1) Configure the learning parameters $(\boldsymbol{\omega}_i, b_i), i = 1, 2, \ldots, L$ randomly, and set $\lambda_0 = 1$;
(2) Calculate $\mathbf{H}_0$ and $\boldsymbol{\beta}_0$ according to Equations (9) and (10);
(3) Set $k = 0$.

Step 2. Sequential learning phase: iteratively train the network using the data chunk at time k+1 $\aleph_{k+1} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=(\sum_{j=0}^{k} N_j)+1}^{\sum_{j=0}^{k+1} N_j}$;

(1) Assess the confidence coefficient of this new data chunk according to the test module in Figure 1, and determine the corresponding weight λ_{k+1};
(2) Calculate the partial $\mathbf{h}_{k+1}$ and $\mathbf{t}_{k+1}$as Equations (11) and (12);
(3) Compute $\boldsymbol{\beta}'_{k+1}$ in an iterative way in accordance with Equations (21)–(23);
(4) Set $k = k + 1$ and go to Step 2 until all the training data chunks are used for the learning process.

Remark 1. *Actually, the AWOS-ELM is the OS-ELM algorithm with adaptive weight. As a new data chunk comes, the proposed algorithm need not repeat the training process of ELM. The AWOS-ELM uses the newly arriving data chunk and the known information which is learned before to conduct the update of the training network, while the ELM applies all data chunks to update the network parameters. Therefore, in the case of sequential predicting problems, the AWOS-ELM algorithm tends to produce a better training process than the ELM algorithm.*

Remark 2. *In the learning process by AWOS-ELM, since the confidence coefficient of each data chunk varies with time, the weight of each data chunk is assessed as soon as the new training data arrives at the next unit time and the SLFN will be trained accordingly. Therefore, the learning process can aptly deal with the confidence coefficient of each data chunk.*

Remark 3. *If $\lambda_0 = \lambda_1 = \cdots = \lambda_k = \cdots$, that is, each training data chunk has the same assessed weight, then it is obvious that AWOS-ELM is equivalent to OS-ELM, indicating that the OS-ELM is a special case of the AWOS-ELM algorithm.*

4. Experiments

For the purpose of verifying the validity of the proposed AWOS-ELM algorithm, six benchmark data sets and an identification problem on the dynamics of a flexible robot arm are considered in this section for the performance comparison between the OS-ELM and AWOS-ELM algorithm. The attributes of each dataset are uniformed into the range [−1,1], and the corresponding outputs are uniformed into [0,1]. The software environment for all experiments is MATLAB 7.11 (MathWorks, Natick, MA, USA) and the hardware environment is an ordinary PC with Intel Core i5-3210M processor. With the purpose of obtaining reliable statistical results, fifty trials are carried out for each case. For the purpose of comparing the performance, the definition of the rooted mean squared errors (RMSE) is given as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N_{testing}} \left\| \hat{\mathbf{t}}_i - \mathbf{t}_i \right\|_F^2}{N_{testing} \times m}} \tag{29}$$

where $\hat{\mathbf{t}}_i$ is the prediction value in regard to the target value $\mathbf{t}_i$, $N_{testing}$ denotes the testing samples number. For a learning algorithm, a smaller RMSE often indicates a better generalization performance. In addition, σ, the standard deviation of RMSEs in 50 trials, can effectively reflect the reliability of the proposed algorithm. Two classical hidden node functions, the radial basis function $h(\mathbf{x}) = e^{-b_i \|\mathbf{x} - a_i\|^2}$ and the sigmoid function $h(\mathbf{x}) = \frac{1}{1+e^{-(a_i \cdot \mathbf{x} + b_i)}}$, are chosen for each learning algorithm. If the hidden function is set to be sigmoid, $\boldsymbol{\omega}_i$ and b_i are chosen according to uniform distribution in

the closed interval [−1,1], and if the RBF function is selected, $\boldsymbol{\omega}_i$ is also obtained from the uniform distribution in [−1,1], while b_i is determined according to the uniform distribution in the open interval (0,0.5) [15]. For the AWOS-ELM algorithm, we select the threshold τ_r from the domain $\{x|x = 0.02 + 0.01k, k = 0, 1, 2, \cdots 18\}$, and the best threshold is selected manually according to the prediction performance. In addition, the regulation parameter is set as $C = 10^{-5}$ [24]. In order to verify the prediction performance of AWOS-ELM with the existence of disturbance, Gaussian noise with a standard deviation of 0.0015 is added into the training samples. For the purpose of having a fair comparison, the hidden nodes number of OS-ELM is optimized by the cross validation method, afterwards the optimized hidden nodes number is extended to the AWOS-ELM algorithm.

4.1. Benchmark Data Sets

In this subsection, we make a comparison between AWOS-ELM and OS-ELM on six time series prediction problems consisting of three artificial time series and three actual time series. Among them, a monthly milk production dataset, having 100 training and 44 testing data, and an electricity production dataset, having 350 training and 102 testing data, are obtained from the well-known Time Series Data Library. The sunspot time series, having 2500 training and 690 testing data, is the monthly mean total sunspot number from January 1749 to December 2015 and is obtained from Solar Influences Data Analysis Center. Pseudo periodic synthetic time series, having 8000 training and 1801 testing data, is found from the University of California, Irvine (UCI) repository. The other two chaotic series, viz. Mackey-Glass and Logistic time series, are generated according to mathematical equations. The Mackey-Glass series $\left\{x_k^{mg}|k = 1, 2, 3, \ldots\right\}$ is generated according to the following differential delay equation [25]

$$\frac{dx^{mg}(t)}{dt} = \frac{a(t-\tau)}{1 + x^{mg}(t-\tau)} - bx^{mg}(t) \tag{30}$$

where $\tau = 17$, $a = 0.2$, $b = 0.1$, and $x(0) = 1.2$. The Logistic series $\left\{x_k^{lo}|k = 1, 2, 3, \ldots\right\}$ is produced by the following recursive equation [26],

$$x_{k+1}^{lo} = \lambda x_k^{lo}\left(1 - x_k^{lo}\right) \tag{31}$$

where $\lambda = 3.5$. The prediction performance comparison between AWOS-ELM and OS-ELM on benchmark data sets is given in Table 1. The mean and standard deviation of weights for AWOS-ELM algorithm are listed in Table 2. The results in Tables 1 and 2 are the mean values of 50 trials.

As observed from Table 1, the training time for AWOS-ELM is close to that for OS-ELM in various data sets, just as we expected. When the hidden nodes number and chunk size are set to be consistent, AWOS-ELM has lower RMSE and lower standard deviation than OS-ELM in most data sets. This implies higher prediction accuracy and the superior prediction stability of AWOS-ELM algorithm due to the accessed confidence coefficient. Furthermore, from Table 2, we can find that the mean weights are less than 0.92, which implies that the proportion of low confidence data is considerable. Additionally, the standard deviations are quite large, which implies that there is a great difference between the weights of normal data and that of low confidence data in the AWOS-ELM algorithm. Thus, the AWOS-ELM can efficiently increase the prediction performance.

Table 1. Performance comparison between AWOS-ELM and OS-ELM on benchmark data sets. RMSE, rooted mean squared errors.

Data Sets	Hidden Node Type	Algorithms	RMSE	σ	Hidden Nodes Number	τ_r	Chunk Size	Training Time/s
Logistic	Sigmoid function	OS-ELM	0.0164	0.0046	50	-	10	0.0924
		AWOS-ELM	0.0041	0.0010	50	0.09	10	0.0914
	Radial basis function	OS-ELM	0.0099	0.0023	50	-	10	0.1919
		AWOS-ELM	0.0039	0.0007	50	0.09	10	0.1913
Mackey-Glass	Sigmoid function	OS-ELM	0.0370	0.0021	25	-	5	0.1638
		AWOS-ELM	0.0234	0.0026	25	0.1	5	0.1675
	Radial basis function	OS-ELM	0.0362	0.0028	25	-	5	0.2764
		AWOS-ELM	0.0199	0.0024	25	0.1	5	0.2845
Sunspot	Sigmoid function	OS-ELM	0.0859	0.0006	40	-	10	0.0612
		AWOS-ELM	0.0833	0.0006	40	0.15	10	0.0596
	Radial basis function	OS-ELM	0.0844	0.0005	40	-	10	0.1251
		AWOS-ELM	0.0831	0.0004	40	0.15	10	0.1236
Pseudo periodic synthetic series	Sigmoid function	OS-ELM	0.0342	0.0007	20	-	15	0.1760
		AWOS-ELM	0.0094	0.0004	20	0.1	15	0.1847
	Radial basis function	OS-ELM	0.0365	0.0016	20	-	15	0.2577
		AWOS-ELM	0.0219	0.0025	20	0.1	15	0.2671
Milk production	Sigmoid function	OS-ELM	0.0561	0.0096	15	-	1	0.0062
		AWOS-ELM	0.0394	0.0110	15	0.12	1	0.0056
	Radial basis function	OS-ELM	0.0638	0.0101	15	-	1	0.0125
		AWOS-ELM	0.0506	0.0097	15	0.12	1	0.0103
Electricity production	Sigmoid function	OS-ELM	0.0301	0.0053	12	-	1	0.0168
		AWOS-ELM	0.0265	0.0034	12	0.08	1	0.0165
	Radial basis function	OS-ELM	0.0566	0.0233	12	-	1	0.0315
		AWOS-ELM	0.0406	0.0124	12	0.08	1	0.0303

Table 2. Mean and standard deviation of the adaptive weights for AWOS-ELM algorithm.

Data Sets	Sigmoid Hidden Node		Radial Basis Function Hidden Node	
	Mean	Standard Deviation	Mean	Standard Deviation
Logistic	0.7366	0.4403	0.7855	0.4080
Mackey-Glass	0.8750	0.3311	0.8748	0.3313
Sunspot	0.6440	0.4729	0.6447	0.4726
Pseudo periodic synthetic series	0.8726	0.3338	0.8592	0.3476
Milk production	0.8615	0.3435	0.9009	0.2959
Electricity production	0.8857	0.3152	0.9176	0.2741

4.2. Robot Arm Example

In this subsection, the performance of AWOS-ELM and OS-ELM are compared on the identification problem regarding the dynamics modeling of a robot arm. The system input and output of this flexible robot arm are the measured reaction torque and the acceleration, respectively [27]. There are two attributes in this flexible robot arm example, and all of the 1024 pairs of data are described in Figure 2. In order to learn the model, the input and output of SLFN, $\mathbf{x}_i$ and $\mathbf{t}_i$, are respectively defined as $\mathbf{x}_i = [u_i, u_{i-1}, u_{i-2}, u_{i-3}, u_{i-4}, d_{i-1}, d_{i-2}, d_{i-3}, d_{i-4}]^T$ and $\mathbf{t}_i = d_i$, where u_i and d_i are the measured reaction torque and the acceleration of this robot arm, respectively. Thus, the samples number is 1019. The training set contains the front 819 samples, and the rest of the 200 samples are used to test the network.

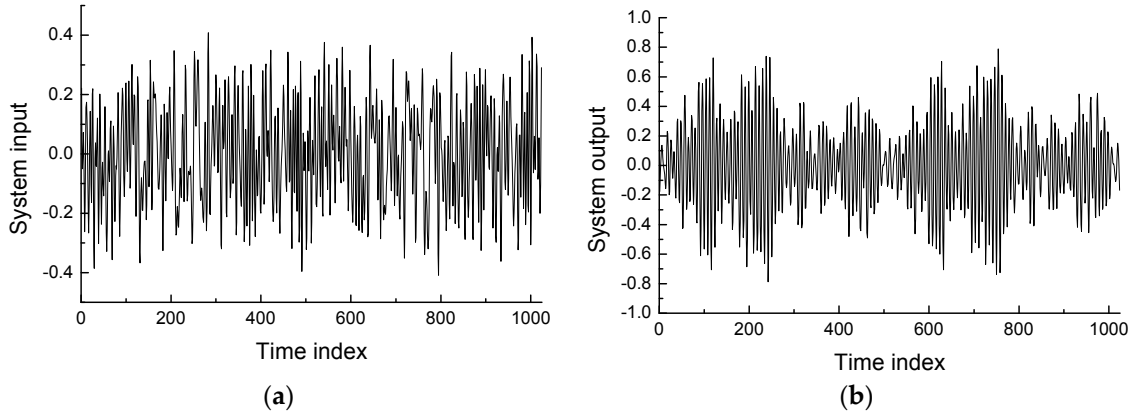

Figure 2. Flexible robot arm attributes. (**a**) System input; (**b**) system output.

Figure 3 illustrates the relationship between the hidden nodes number and testing RMSE of the OS-ELM and AWOS-ELM algorithms. It shows that the OS-ELM and AWOS-ELM with low hidden nodes number have similar testing RMSE, and the testing RMSE of AWOS-ELM algorithm tends to be lower than that of OS-ELM when the hidden nodes number increases. It shows that the OS-ELM and AWOS-ELM with low hidden nodes number have similar testing RMSE, and the testing RMSE of the AWOS-ELM algorithm tends to be lower than that of OS-ELM when the hidden nodes number increases. When the number of hidden nodes is too small, the prediction accuracy of SLFN trained by the OS-ELM or AWOS-ELM algorithm is very low. As described in Equations (17) and (18), the computing adaptive weight depends on the prediction value. If the prediction accuracy is too low, the AWOS-ELM cannot differentiate the low confidence data and normal data very well. Thus, in Figure 3, the performance of AWOS-ELM is similar to or even slightly less than that of OS-ELM when the hidden node number is low. In order to be fair for the sake of comparison, the hidden nodes number is set to be 45, which can guarantee that the algorithms with sigmoid function or RBF function both have fine performance levels. Figure 4 illustrates the identification effectiveness of the flexible robot arm dynamics modeling problem, and we can easily find that the AWOS-ELM algorithm has higher prediction accuracy than OS-ELM algorithm no matter whether considering the sigmoid function case or RBF function case. Table 3 showcases that the proposed AWOS-ELM algorithm and OS-ELM algorithm have similar training time and standard deviation values, implying the similar learning speed and stability, and that the AWOS-ELM algorithm has lower RMSE than the OS-ELM algorithm, which implies better generalization performance and higher prediction accuracy.

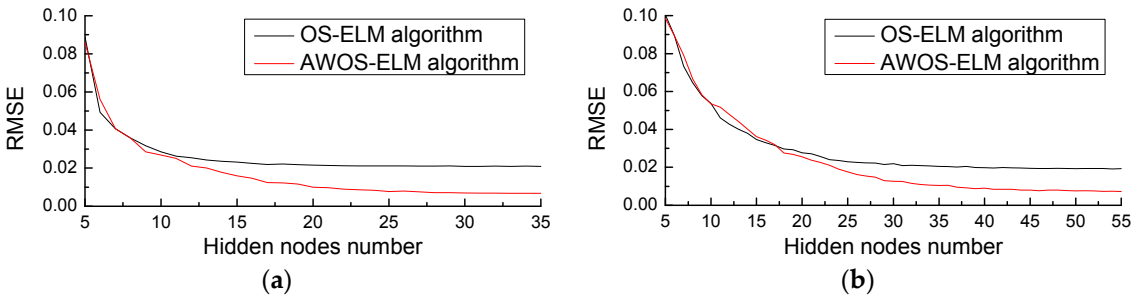

Figure 3. Relationship between testing RMSE and hidden nodes number. (**a**) Sigmoid node; (**b**) RBF node.

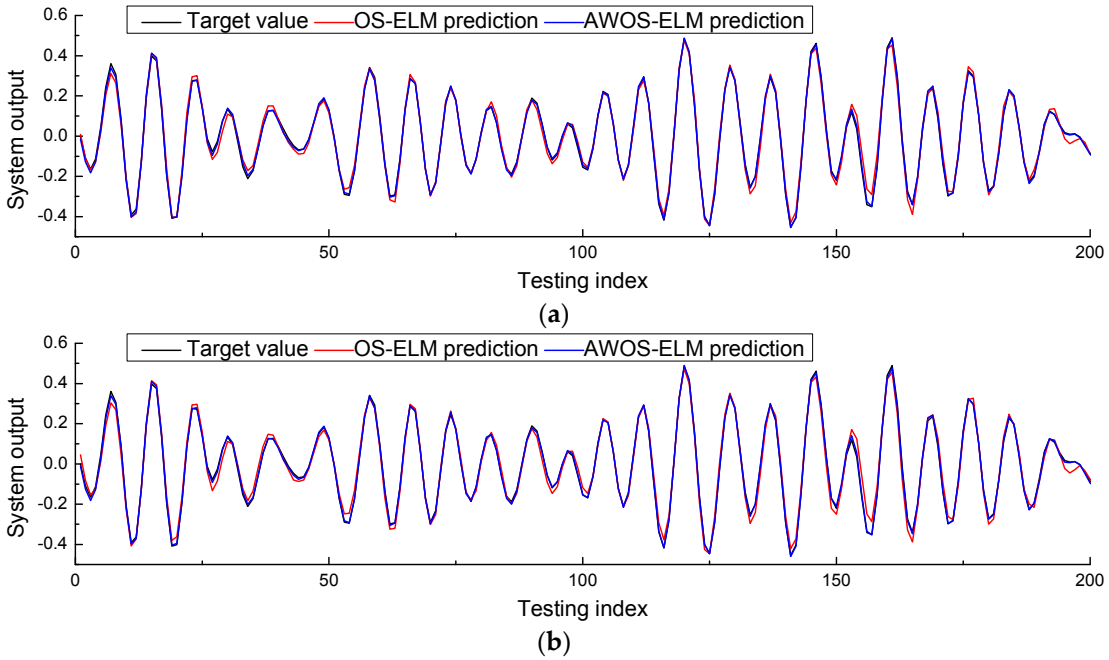

Figure 4. Comparison of the prediction ability of the OS-ELM algorithm and the AWOS-ELM algorithm. (**a**) Sigmoid function; (**b**) RBF function.

Table 3. Performance comparison between AWOS-ELM and OS-ELM on the robot arm example.

Hidden Node Type	Algorithms	RMSE	σ	Hidden Nodes Number	τ_r	Chunk Size	Training Time/s
Sigmoid function	OS-ELM	0.0207	0.0004	45	-	5	0.1401
	AWOS-ELM	0.0081	0.0004	45	0.04	5	0.1426
Radial basis function	OS-ELM	0.0198	0.0006	45	-	5	0.3738
	AWOS-ELM	0.0084	0.0007	45	0.04	5	0.4115

5. Conclusions

In lots of online learning applications, the sequentially arrived data usually have varying confidence coefficients. OS-ELM trains the neural network chunk-by-chunk, but at the same time, it cannot deal with varying confidence coefficients of each data chunk very well. Based on the OS-ELM algorithm, we propose a novel algorithm, AWOS-ELM, which assesses the confidence coefficient and determines the weight of each data chunk before using the chunk to train the network. The proposed AWOS-ELM algorithm can improve the learning process by picking out the abnormal samples which may mislead the subsequent learning process, thereby reducing the impact of these abnormal samples. Thus, the AWOS-ELM is able to learn sequentially similar to how the OS-ELM works, but at the same time deals with the varying confidence coefficients of each data chunk properly. Compared with OS-ELM, simulations on benchmark databases demonstrate that AWOS-LEM performs better in generalization performance, stability, and prediction accuracy. In addition, experimental results on a flexible robot arm identification problem demonstrate that AWOS-ELM has the ability to get higher prediction accuracy than OS-ELM, while the two algorithms have similar training speed and stability.

Acknowledgments: This research was funded by the National Natural Science Foundation of China (under Grant 51276087).

Author Contributions: Jinquan Huang and Feng Lu conceived and designed the main idea, Junjie Lu wrote the program and carried out the experiments, Junjie Lu and Jinquan Huang analyzed the data and interpreted the results, Junjie Lu and Feng Lu wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shukla, A.K.; Garde, Y.A.; Jain, I. Forecast of Weather Parameters Using Time Series Data. *MAUSAM* **2014**, *65*, 209–520.
2. Kato, R.; Nagao, T. Stock Market Prediction Based On Interrelated Time Series Data. In Proceedings of the IEEE Symposium on Computers & Informatics, Penang, Malaysia, 18–20 March 2012.
3. Soni, S.K.; Chand, N.; Singh, D.P. Reducing the Data Transmission in WSNs Using Time Series Prediction Model. In Proceedings of the IEEE International Conference on Signal Processing, Communications and Computing, Hong Kong, China, 13–15 August 2012.
4. Hulsmann, M.; Borscheid, D.; Friedrich, C.M.; Reith, D. General Sales Forecast Models for Automobile Markets and Their Analysis. *Trans. Mach. Learn. Data Min.* **2011**, *5*, 65–86.
5. Gooijer, J.G.D.; Hyndman, R.J. 25 Years of Time Series Forecasting. *Int. J. Forecast.* **2006**, *22*, 443–473. [CrossRef]
6. Box, G.E.P.; Jenkins, S.B. Time Series Analysis: Forecasting and Control. *J. Oper. Res. Soc.* **1976**, *22*, 199–201.
7. Smith, C.; Jin, Y. Evolutionary Multi-objective Generation of Recurrent Neural Network Ensembles for Time Series Prediction. *Neurocomputing* **2014**, *143*, 302–311. [CrossRef]
8. Donmanska, D.; Wojtylak, M. Application of Fuzzy Time Series Models for Forecasting Pollution Concentrations. *Expert Syst. Appl.* **2012**, *39*, 7673–7679. [CrossRef]
9. Kumar, N.; Jha, G.K. A Time Series ANN Approach for Weather Forecasting. *Int. J. Control Theory Comput. Model* **2013**, *3*, 19–25. [CrossRef]
10. Claveria, O.; Torra, S. Forecasting Tourism Demand to Catalonia: Neural Networks vs. Time Series Models. *Econ. Model* **2014**, *36*, 220–228. [CrossRef]
11. Flores, J.J.; Graff, M.; Rodriguez, H. Evolutive Design of ARMA and ANN Models for Time Series Forecasting. *Renew. Energ.* **2012**, *44*, 225–230. [CrossRef]
12. Adhikari, R. A Neural Network Based Linear Ensemble Framework for Time Series Forecasting. *Neurocomputing* **2015**, *157*, 231–242. [CrossRef]
13. Kumar, D.A.; Murugan, S. Performance Analysis of Indian Stock Market Index Using Neural Network Time Series Model. In Proceedings of the International Conference on Pattern Recognition, Informatics and Mobile Engineering, Salem, India, 21–22 February 2013.
14. Yoon, H.; Hyun, Y.; Ha, K.; Lee, K.; Kim, G. A Method to Improve the Stability and Accuracy of ANN- and SVM-Based Time Series Models For Long-Term Groundwater Level Predictions. *Comput. Geosci.* **2016**, *90*, 144–155. [CrossRef]
15. Huang, G.B.; Chen, L.; Siew, C.K. Universal Approximation Using Incremental Constructive Feedforward Networks with Random Hidden Nodes. *IEEE T. Neural Netw.* **2006**, *17*, 879–892. [CrossRef] [PubMed]
16. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. In Proceedings of the International Joint Conference on Neural Networks, Budapest, Hungary, 25–29 July 2004.
17. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme Learning Machine: Theory and Applications. *Neurocomputing.* **2006**, *70*, 489–501. [CrossRef]
18. Hosseinioun, N. Forecasting Outlier Occurrence in Stock Market Time Series Based on Wavelet Transform and Adaptive ELM Algorithm. *J. Math. Financ.* **2016**, *6*, 127–133. [CrossRef]
19. Dash, R.; Dash, P.K.; Bisoi, R. A Self Adaptive Differential Harmony Search Based Optimized Extreme Learning Machine for Financial Time Series Prediction. *Swarm Evol. Comput.* **2014**, *19*, 25–42. [CrossRef]
20. Liang, N.Y.; Huang, G.B.; Saratchandran, P.; Sundararajan, N. A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks. *IEEE T. Neural Netw.* **2006**, *17*, 1411–1423. [CrossRef] [PubMed]
21. Huang, G.B.; Zhou, H.M.; Ding, X.J.; Zhang, R. Extreme Learning Machine for Regression and Multiclass Classification. *IEEE Trans. Syst. Man Cybern. B* **2012**, *42*, 513–529. [CrossRef] [PubMed]
22. Huang, G.; Ding, X.; Zhou, H. Optimization Method Based Extreme Learning Machine for Classification. *Neurocomputing.* **2010**, *74*, 155–163. [CrossRef]
23. Deng, C.Y. A Generalization of the Sherman-Morrison-Woodbury Formula. *Appl. Math. Lett.* **2011**, *24*, 1561–1564. [CrossRef]

24. Guo, W.; Xu, T.; Tang, K. M-Estimator-Based Online Sequential Extreme Learning Machine For Predicting Chaotic Time Series with Outliers. *Neural Comput. Appl.* **2016**, *2016*, 1–18. [CrossRef]
25. EI-Sayed, A.M.; Salman, S.M.; Elabd, N.A. On a Fractional-Order Delay Mackey-Glass Equation. *Adv. Differ. Equ.* **2016**, *2016*, 1–11.
26. Berezowski, M.; Grabski, A. Chaotic and Non-chaotic Mixed Oscillations in a Logistic Systems with Delay. *Chaos Solitons Fractals* **2002**, *14*, 1–6. [CrossRef]
27. Balasundaram, S.; Kapil, D.G. Lagrangian Support Vector Regression via Unconstrained Convex Minimization. *Neural Net.* **2014**, *51*, 67–79. [CrossRef] [PubMed]