Hindawi
Complexity
Volume 2020, Article ID 3284587, 14 pages
https://doi.org/10.1155/2020/3284587

Research Article

Estimating Weak Pulse Signal in Chaotic Background with Jordan Neural Network

Liyun Su and Xiu Ling

School of Science, Chongqing University of Technology, Chongqing 400054, China

Correspondence should be addressed to Liyun Su; cloudhopping@163.com

Received 6 February 2020; Revised 19 May 2020; Accepted 17 June 2020; Published 20 July 2020

Academic Editor: Mohamed Boutayeb

In target estimating sea clutter or actual mechanical fault diagnosis, useful signal is often submerged in strong chaotic noise, and the targeted signal data are difficult to recover. Traditional schemes, such as Elman neural network (ENN), backpropagation neural network (BPNN), support vector machine (SVM), and multilayer perceptron- (MLP-) based model, are insufficient to extract the weak signal embedded in a chaotic background. To improve the estimating accuracy, a novel estimating method for aiming at extracting problem of weak pulse signal buried in a strong chaotic background is presented. Firstly, the proposed method obtains the vector sequence signal by reconstructing higher-dimensional phase space data matrix according to the Takens theorem. Then, a Jordan neural network- (JNN-) based model is designed, which can minimize the error squared sum by mixing the single-point jump model for targeting signal. Finally, based on short-term predictability of chaotic background, estimation of weak pulse signal from the chaotic background is achieved by a profile least square method for optimizing the proposed model parameters. The data generated by the Lorenz system are used as chaotic background noise for the simulation experiment. The simulation results show that Jordan neural network and profile least square algorithm are effective in estimating weak pulse signal from chaotic background. Compared with the traditional method, (1) the presented method can estimate the weak pulse signal in strong chaotic noise under lower error than ENN-based, BPNN-based, SVM-based, and -ased models and (2) the proposed method can extract the weak pulse signal under a higher output SNR than BPNN-based model.

1. Introduction

As early as the end of the 19th century, the French Poincare found that the solution of the three-body problem could be random within a certain range when he studied the three-body problem, but this discovery did not arouse widespread concern among scholars. It was not until 1963 that Lorenz, an American meteorologist, published a paper [1] that gave the initial condition parameters for generating chaos in differential equations, which were called Lorenz equations. With the continuous in-depth study of chaos, the characteristics and scientific concepts of chaos gradually come into people's vision. Chaotic phenomena can be described as random phenomena in a deterministic system, which is a complex behavior generated by a nonlinear dynamic system. Moreover, chaos has three characteristics: sensitivity to initial value, short-term predictability, and intrinsic randomness.

Pulse signal is a kind of discrete signal, which is usually submerged by noise and not easy to detect. It is widely existed in communication engineering, biomedicine, and industrial automatic control. Therefore, the prediction of the pulse signal from chaotic noise has certain practical significance. Experts and scholars at home and abroad have also studied the prediction of the pulse signal in chaotic noise and obtained a lot of research results, such as Volterra filter, correlation detection in time domain, spectral analysis in frequency domain, chaotic Duffing oscillator [2], local linear [3–5], multiple kernel extreme learning machine [6], and Kalman filter , extend Kalman filter (EKF) [7], wavelet transform and nonlinear autoregressive model, etc.

At present, experts and scholars have proposed many methods to predict chaotic time series and extract impulse signal from chaotic noise [8, 9]. Jiang Xiangcheng proposed the Volterra adaptive method to predict chaotic time series [10]. The improved local projection algorithm proposed by

Han Min is a nonlinear chaotic noise reduction method, which combines wavelet analysis with local projection to estimate noisy chaotic signals [11]. Wang Shiyuan et al. proposed the fractional order maximum correlation entropy algorithm [12].

Neural network is a model that imitates the information processing mode of the human nervous system. It is the abstraction and simplification of human neural network. Neural networks can approximate and model the processes of nonlinear dynamic systems [13]. Neural network is divided into forward network, feedback network, random neural network, and competitive network. Because of the rich diversity of neural networks, different neural network to build models of nonlinear systems is also being studied. For example, radial quantity function (RBF) [14, 15], support vector machine (SVM) [16], least squares support vector machine (LS-SVM) [17], backpropagation neural network (BPNN) [13], fuzzy neural network (FNN) [18], recursive neural network (RNN) [19, 20], multilayer perceptron (MLP), nonlinear regression model (NAR), recursive predictor (RPNN), and ESN [21]. Chandra used the method of memetic cooperative coevolution to train the Elman recurrent neural network on the problem of grammatical inference [22]. Two years later, Chandra used Elman neural network to characterize chaotic time series and added cooperation and competition into them, so as to achieve the purpose of prediction [20]. This paper uses the Jordan network, a neural network established by Jordan in 1986 [23]. Both the Jordan neural network and the Elman neural network (ENN) are cyclic neural networks. The difference is that the Elman neural network feeds the output value of the hidden layer at the previous moment back to the feedback layer, while the Jordan neural network feeds the output value of the output layer at the previous moment back to the feedback layer, which can be used to represent the dynamic system.

In this paper, the Jordan neural network is used to fit the chaotic background and estimate the pulse signal from its residuals. Because the pulse signal is very weak, the characteristics of chaotic time series are mainly reflected in the chaotic background. Firstly, the observed signal was reconstructed, then the Jordan neural network and the single-point pulse signal model were established, the weight and bias of the Jordan neural network were optimized by the gradient descent algorithm, and then the amplitude of the pulse signal was estimated by the profile least square method.

The remainder of the paper is organized as follows. Section 2 gives a brief background and related work. Section 3 presents and analyzes this problem. Section 4 estimates the weak pulse signal from a chaotic background by Jordan neural network. Section 5 presents 3 experimental simulations results and discussion. Finally, conclusions and discussion on future work are provided in Section 6.

2. Background and Related Work

Our work mainly involves two aspects: the application of feedback neural network and the application of feedback neural network in time series prediction.

2.1. Application of Feedback Neural Network. Chaotic time series is a kind of irregular motion in deterministic system and a special time series. However, the traditional time series prediction models—MA, ARMA, and ARIMA—and other models are not ideal for chaotic time series prediction.

The applications for chaotic time series prediction are wide and range from financial prediction to weather prediction [20].

Recursive neural networks are used in a wide range of applications, such as weather prediction [20], unbalanced fault diagnosis [24], and detection of network malware [25]. Classification is also an important application of recursive neural network. Recently, recursive neural networks have been widely integrated into convolutional neural networks. Epstein integrates the recursive neural network into the convolutional neural network for image classification. Experiments show that the model is more robust to category-independent attributes [26]. Not only is the wide application range of neural network reflected in its high frequency of use in different fields but also it can be flexibly combined with other networks, models, and methods. Slawek Smyl found that the preprocessing parameters were actually updated formulas from some models of the exponential smoothing family. Therefore, it proposes a hybrid prediction method in [27], which mixes the exponential smoothing model with the advanced long and short-term memory neural network in a common framework. The advantages of this hybrid prediction method are that the exponential smoothing equation enables the method to effectively capture the major components of a single sequence, while neural networks allow for nonlinear trending and cross-learning.

2.2. Application of Feedback Neural Network in Time-Series Prediction. Liu et al. combined the dual stage two-phase model with the feedback neural network to make a long-term prediction of multiple time series in [28]. And it has been successfully used to develop widely used expert or intelligent systems. In [29], Wei Wu used ARIMA model, Elman neural network, and Jordan neural network, respectively, to predict the nonlinear time series data of human brucellosis in China, and the results showed that the feedback neural network was more accurate than the traditional ARIMA model. In [30], Zhihao Wang proposed a heart sound signal recovery method based on the long and short time memory prediction model based on the structure of circular neuron network. This method can not only restore incomplete or disturbed signals, but also have filtering effect. The damaged or incomplete heart sound signal has a good recovery effect, which has a promoting effect on medical research. Zhang proposed a new nonstationary time series modeling optimization algorithm in [31]. The algorithm uses moving window and exponential decay weight to eliminate the influence of historical gradient. The regretful boundary of the new algorithm is analyzed to ensure the convergence of the calculation. The simulation results of this algorithm on the data set of short-term power load are better than the existing optimization algorithm.

3. Problem Analysis

This paper analyzes the weak pulse signal which is fixed chaotic noise. Let t be a time parameter. We can assume that observation sequence, expressed as $x(t)$, contains chaotic noise $c(t)$, white noise signal $n(t)$, and weak pulse signal $s(t)$. So we have a formula as follows:

$$x(t) = s(t) + c(t) + n(t). \tag{1}$$

Pulse signal can be boiled down by periodic signal:

$$s(t) = \sum_{i=1}^{k} a_i s_i(t) = a^{\mathrm{T}} s(t), \tag{2}$$

where

$$\begin{aligned} a &= (a_1, a_2, \cdots, a_k)^{\mathrm{T}}, \\ s(t) &= (s_1(t), s_2(t), \cdots, s_k(t))^{\mathrm{T}}, \\ s_i(t) &= \begin{cases} 1, & t = T_i, 2T_i \cdots \\ 0, & \text{others}, \end{cases} \end{aligned} \tag{3}$$

where $a_i (i = 1, 2, \cdots, k)$ is the amplitude of the weak pulse signal $s_i(t)(i = 1, 2, \cdots, k)$ and T_i is the periodic of $s_i(t)(i = 1, 2, \cdots, k)$. Estimating weak pulse signal from $x(t)$, we can consider that estimating weak pulse signal $s(t)$ in chaotic background is to estimate the value of a. Then, our ultimate goal turns to estimating a.

4. Estimating the Pulse Signal from Strong Chaotic Background

4.1. Phase Space Reconstruction. Chaotic time series is a univariate or multivariable time series produced by a dynamical system. Phase space reconstruction theorem is mapping a chaotic time series to high-dimensional space. According to the Takens theorem of embedding [32], there is a correlation between one component of the dynamical system and the other components.

For an observed chaotic signal $\{x(t), t = 1, 2, \cdots, n\}$, the embedding dimension m and the time delay τ could be obtained by using Cao's method [33], and a phase point in the reconstructed phase space can be expressed as $X(t) = (x(t), x(t-\tau), \cdots, x(t-(m-1)\tau))^{\mathrm{T}}$, where $t = n_1, n_1 + 1, \cdots, n$ and $n_1 = (m-1)\tau + 1$. Then there is a smooth mapping, g: $\mathrm{R}^m \longrightarrow \mathrm{R}$, which provides the relationship between $X(t)$ and $y(t)$, denoted as $y(t) = g(X(t))(t = n_1, n_1 + 1, \cdots, n-1)$. To predict chaotic time series is to find the function g or the approximate function of g.

Remark 1. In mathematics, Takens embedding theorem gives the conditions under which a chaotic dynamical system can be reconstructed from a sequence of observations of the state of a dynamical system. The reconstruction preserves the properties of the dynamical system that do not change under smooth coordinate changes, but it does not preserve the geometric shape of structures in phase space. It provides the conditions under which a smooth attractor can be reconstructed from the observations made with a generic function.

Cao's method is a practical method which determines the minimum embedding dimension from a scalar time series. The average ratio of Euclidean distance of a vector with embedded dimension m to Euclidean distance with embedded dimension $m+1$ is denoted as $\mathrm{E}(m)$. The ratio of $\mathrm{E}(m+1)$ to $\mathrm{E}(m)$ stops changing when m is greater than some value m_0 if the time series comes from an attractor. Then $m_0 + 1$ is the minimum embedding dimension we look for.

4.2. Jordan Neural Network. Jordan combined the Hopfield network storage concept and distributed parallel processing theory to create a new circular neural network in 1986 [23]. Jordan neural network is made of 4 parts: input layer, hidden layer, output layer, and context layer, respectively. There is a first-order delay operator in the connection from the output layer to the context layer, so that the context layer stores the information of the output layer. There are two types of activation functions in the neural network: linear activation function and nonlinear activation function. Sigmoid nonlinear activation function which is expressed as $f(x) = 1/1 + e^{-x}$ is used at the hidden layer and the linear function is used at the output layer in this paper. Jordan neural network topology is shown in Figure 1.

According to Figure 1, $X(t) = (x(t), x(t-\tau), x(t-2\tau), \cdots, x(t-(m-1)\tau))^{\mathrm{T}}$ is used to represent the input vector of the Jordan neural network, where $t = n_1, n_1 + 1, \cdots, n$ and $n_1 = 1 + (m-1)\tau$. $h(t) = (h_1(t), h_2(t), \cdots, h_r(t))^{\mathrm{T}}$ is the output vector of hidden layer. $(W_{ij}^{(1)})(i = 1, 2, \cdots, r, j = 1, 2, \cdots, m)$ are the weights from input layer neuron (j) to ith first hidden layer neuron. $(W_i^{(2)})(i = 1, 2, \cdots, r)$ are the weights from context layer neuron to ith first hidden layer neuron. $(W_i^{(3)})(i = 1, 2, \cdots, r)$ are the weights from hidden layer neuron (i) to output layer neuron. $(b_i^{(1)})(i = 1, 2, \cdots, r)$ are the biases of hidden layer neuron (i). $b^{(2)}$ and $b^{(3)}$ are the biases of output layer and context layer, respectively. $f^{(2)}$ and $f^{(3)}$ are the activation function of output layer and context layer, respectively, and they are usually linear function, while $f_i^{(1)}(i = 1, 2, \cdots, r)$ are the activation function of hidden layer neuron (i). $h_i(t)(i = 1, 2, \cdots, r)$ is the value of hidden layer neuron (i). $d(t)$ and $y(t)$ are the value of context layer and output layer.

In order to express clearly and write easily, the following signs will be introduced:

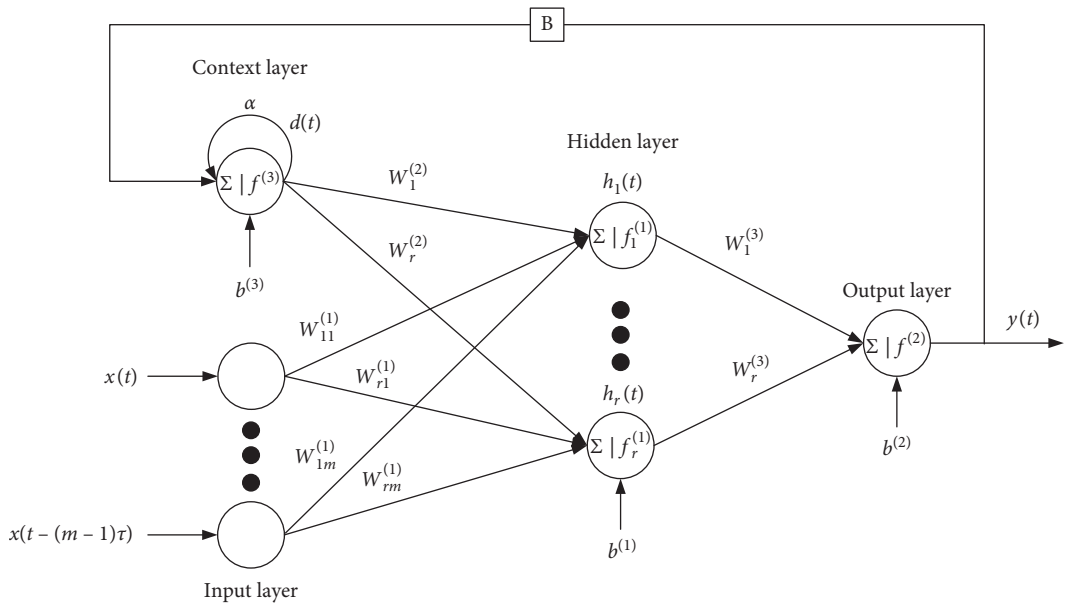

Figure 1: Jordan neural network topology.

$$
\begin{aligned}
W^{(1)} &= \left(W_{ij}^{(1)}\right) = \begin{pmatrix} W_1^{(1)} \\ W_2^{(1)} \\ \vdots \\ W_r^{(1)} \end{pmatrix}, \quad i = 1, 2, \cdots, r;\ j = 1, 2, \cdots, m, \\
W^{(2)} &= \left(W_i^{(2)}\right)_{i=1,2,\cdots,r} = \left(W_1^{(2)}, W_2^{(2)}, \cdots, W_r^{(2)}\right)^{\mathrm{T}}, \\
W^{(3)} &= \left(W_i^{(3)}\right)_{i=1,2,\cdots,r} = \left(W_1^{(3)}, W_2^{(3)}, \cdots, W_r^{(3)}\right), \\
b^{(1)} &= \left(b_1^{(1)}, b_2^{(1)}, \cdots, b_r^{(1)}\right)^{\mathrm{T}}.
\end{aligned} \tag{4}
$$

So the Jordan neural network is obtained:

$$
\begin{aligned}
h_i(t) &= f_i^{(1)}\left(W_i^{(1)} X(t) + W_i^{(2)} d(t) + b_i^{(1)}\right), \quad 1 \le t \le T, i = 1, 2, \cdots r, \\
y(t) &= f^{(2)}\left(\sum_{i=1}^{r} W_i^{(3)} h_i(t) + b^{(2)}\right), \quad 1 \le t \le T, \\
d(t) &= f^{(3)}\left(\alpha\, d(t-1) + y(t-1) + b^{(3)}\right), \quad 1 \le t \le T, \\
d(0) &= d(1) = 0.
\end{aligned} \tag{5}
$$

To better understand equation (5), we provide a detailed diagram of the first three steps of the Jordan neural network in Figure 2.

The following equation is obtained by the Takens theorem of embedding:

$$
\hat{x}(t+1) \approx y(t) = g(X(t)), \tag{6}
$$

where $\hat{x}(t+1)$ is the fitting value of the Jordan neural network at time t. And the smooth mapping g can be expressed as in

$$
g(X(t)) = f^{(2)}\left(\sum_{i=1}^{r}\left[W_i^{(3)} f_i^{(1)}\left(W_i^{(1)} X(t) + W_i^{(2)} f^{(3)}\left(\alpha\, d(t-1) + y(t-1) + b^{(3)}\right) + b_i^{(1)}\right)\right] + b^{(2)}\right). \tag{7}
$$

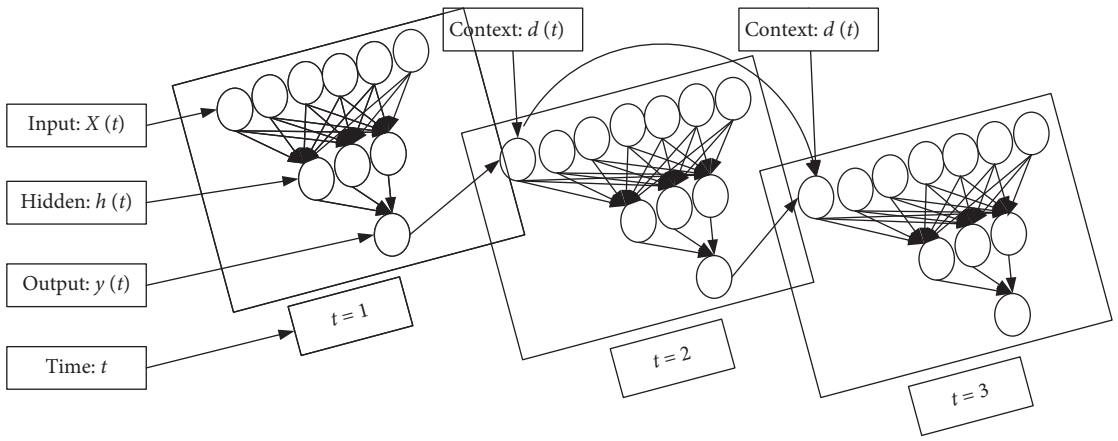

Figure 2: Details of Jordan neural network recurrent connections.

4.3. Model for Estimating the Pulse Signal. According to the Takens theorem of embedding, the model of pulse signal, and Jordan neural network, we can obtain the following equation for estimating the pulse signal:

$$x(t) = s(t) + c(t) + n(t) = a^{\mathrm{T}} s(t) + c(t) + n(t), \tag{8}$$

$$\begin{cases} \tilde{x}(t) = x(t) - a^{\mathrm{T}} s(t) = c(t) + n(t), \\ \tilde{x}(t+1) = g(\tilde{X}(t)) + \varepsilon(t) = g(\tilde{x}(t), \tilde{x}(t-\tau), \cdots, \tilde{x}(t-(m-1)\tau)) + \varepsilon(t). \end{cases} \tag{9}$$

Then, we can deduce the following equation with unknown parameters involving only $g(\cdot)$ and a:

$$x(t+1) - a^{\mathrm{T}} s(t+1) = g\left(x(t) - a^{\mathrm{T}} s(t), \cdots, x(t-(m-1)\tau) - a^{\mathrm{T}} s(t-(m-1)\tau)\right) + \varepsilon(t). \tag{10}$$

Profile least-square method [34] is a basic idea for fitting a semiparametric model. It is semiparametrically efficient for parametric components, and the nonparametric components are fitted if the parametric components were known. Details of the profile least-square estimation are given in the next section.

4.3.1. Profile Least Square Algorithm. Profile least square algorithm is usually divided into two steps: first, estimating the nonparametric function with a given a, resulting in $\hat{g}(\tilde{X}(t))$ and second, estimating the unknown parameter a with the estimated function $\hat{g}(\tilde{X}(t))$.

(A) *Estimation of $g(\tilde{X}(t))$ with Given a.* From equation (10), given the initial value a (null vector), the nonparametric function $\hat{g}(\tilde{X}(t))$ is obtained by minimizing the residual of fitting:

$$\hat{g}(\tilde{X}(t)) = \arg\min_{t} \sum_{t=n_1}^{n-1} [\tilde{x}(t+1) - g(\tilde{X}(t))]^2,$$
$$E = \frac{1}{2} \sum_{t=n_1}^{n-1} [\hat{\tilde{x}}(t+1) - \tilde{x}(t+1)]^2. \tag{11}$$

Gradient descent algorithm is used to update the weights of Jordan neural network to achieve the purpose of optimization. The basic idea of gradient descent algorithm is the linear approximation by first-order Taylor expansion. Therefore, the partial derivatives of the weights and the bias that need to be optimized in this paper are, respectively, calculated as follows:

$$\frac{\partial E}{\partial W^{(3)}} = \sum_{t=n_1}^{n-1} \left\{ \left((\widehat{\widetilde{x}}(t+1) - \widetilde{x}(t+1)) \left(f^{(2)}(\cdot) \right)' h_1(t), \cdots, (\widehat{x}(t+1) - x(t+1)) \left(f^{(2)}(\cdot) \right)' h_r(t) \right) \right\},$$

$$= \sum_{t=n_1}^{n-1} \left\{ (\widehat{\widetilde{x}}(t+1) - \widetilde{x}(t+1)) \left(f^{(2)}(\cdot) \right)' (h_1(t), h_2(t), \cdots, h_r(t)) \right\},$$

$$\frac{\partial E}{\partial W^{(1)}} = \sum_{t=n_1}^{n-1} \left\{ \begin{pmatrix} (\widehat{\widetilde{x}}(t+1) - \widetilde{x}(t+1)) (f^{(2)}(\bullet))' W_1^{(3)} \left(f_1^{(1)}(\bullet) \right)' (X(t))^{\mathrm{T}} \\ \vdots \\ (\widehat{\widetilde{x}}(t+1) - \widetilde{x}(t+1)) (f^{(2)}(\cdot))' W_r^{(3)} (f_r^{(1)}(\cdot))' (X(t))^{\mathrm{T}} \end{pmatrix} \right\},$$

$$\frac{\partial E}{\partial W^{(2)}} = \sum_{t=n_1}^{n-1} \left\{ \begin{pmatrix} (\widehat{\widetilde{x}}(t+1) - \widetilde{x}(t+1)) (f^{(2)}(\cdot))' W_1^{(3)} \left(f_1^{(1)}(\cdot) \right)' d(t) \\ \vdots \\ (\widehat{\widetilde{x}}(t+1) - \widetilde{x}(t+1)) (f^{(2)}(\cdot))' W_r^{(3)} (f_r^{(1)}(\cdot))' d(t) \end{pmatrix} \right\}, \tag{12}$$

$$\frac{\partial E}{\partial b^{(2)}} = \sum_{t=n_1}^{n-1} \left\{ (\widehat{\widetilde{x}}(t+1) - \widetilde{x}(t+1)) \left(f^{(2)}(\cdot) \right)' \right\},$$

$$\frac{\partial E}{\partial b^{(1)}} = \sum_{t-n_1}^{n-1} \left\{ \begin{pmatrix} (\widehat{\widetilde{x}}(t+1) - \widetilde{x}(t+1)) (f^{(2)}(\cdot))' W_1^{(3)} \left(f_1^{(1)}(\cdot) \right)' \\ \vdots \\ (\widehat{\widetilde{x}}(t+1) - \widetilde{x}(t+1)) (f^{(2)}(\cdot))' W_r^{(3)} (f_r^{(1)}(\cdot))' \end{pmatrix} \right\},$$

$$\frac{\partial E}{\partial b^{(3)}} = \sum_{t=n_1}^{n-1} \left\{ \sum_{i=1}^{r} \left((\widehat{\widetilde{x}}(t+1) - \widetilde{x}(t+1)) \left(f^{(2)}(\cdot) \right)' W_i^{(3)} \left(f_i^{(1)}(\cdot) \right)' W_i^{(3)} \left(f^{(3)}(\cdot) \right)' \right) \right\},$$

$$\frac{\partial E}{\partial \alpha} = \sum_{t=n_1}^{n-1} \left\{ \sum_{i=1}^{r} \left((\widehat{\widetilde{x}}(t+1) - \widetilde{x}(t+1)) \left(f^{(2)}(\cdot) \right)' W_i^{(3)} \left(f_i^{(1)}(\cdot) \right)' W_i^{(2)} \left(f^{(3)}(\cdot) \right)' d(t-1) \right) \right\}.$$

The updates of weights and the bias are equal to the sum of the corresponding old one and the product of the partial derivative and the learning rate.

(B) *Estimating a for Given* $\widehat{g}(\widetilde{X}(t))$. For the given $\widehat{g}(\widetilde{X}(t))$, the key to estimate a is to minimize the following equation:

$$a = \arg\min_{a} \sum_{t=n_1}^{n-1} [\widetilde{x}(t+1) - \widehat{g}(\widetilde{X}(t))]^2. \tag{13}$$

The method of estimating a is Newton–Raphson iteration method. Assuming that $R(a) = \sum_{t=n_1}^{n-1} [\widetilde{x}(t+1) - \widehat{g}(\widetilde{X}(t))]^2$ and a_1 is the minimum solution of $R(a)$, so we have $(R(a_1))' = 0$. Then there is the Taylor expansion at the point $a^{(0)}$ near a_1:

$$0 = (R(a_1))' \approx \left(R\left(a^{(0)}\right) \right)' + \left(R\left(a^{(0)}\right) \right)'' \left(a_1 - a^{(0)} \right),$$
$$a^{(1)} = a^{(0)} - \left[\left(R\left(a^{(0)}\right) \right)'' \right]^{-1} \left(R\left(a^{(0)}\right) \right)'. \tag{14}$$

The first derivative and the second derivative of $R(a)$ are obtained with respect to the estimated parameters a:

$$(R(a))' = 2\sum_{t=n_1}^{n-1}\left[(\tilde{x}(t+1) - \hat{g}(\tilde{X}(t)))\left(-s(t+1) - \frac{\partial \hat{g}(\tilde{X}(t))}{\partial a}\right)\right], \tag{15}$$

$$\begin{aligned}(R(a))'' = {} & 2\sum_{t=n_1}^{n-1}\left\{(\tilde{x}(t+1) - \hat{g}(\tilde{X}(t)))\left(-s(t+1) - \frac{\partial \hat{g}(\tilde{X}(t))}{\partial a}\right)\left(-s(t+1) - \frac{\partial \hat{g}(\tilde{X}(t))}{\partial a}\right)^T\right\} \\ & + 2\sum_{t=n_1}^{n-1}\left\{(\tilde{x}(t+1) - \hat{g}(\tilde{X}(t)))\left(-\frac{\partial^2 \hat{g}(\tilde{X}(t))}{\partial a^2}\right)\right\}.\end{aligned} \tag{16}$$

You obtain the following equation from Sections 4.1 through 4.3:

$$\begin{cases}\tilde{X}(t) = X(t) - S(t)a \\ \hat{g}(\tilde{X}(t)) = f^{(2)}\left(\sum_{i=1}^{r}\left[W_i^{(3)} f_i^{(1)}\left(W_i^{(1)}\tilde{X}(t) + W_i^{(2)} f^{(3)}\left(\alpha d(t-1) + y(t-1) + b^{(3)}\right) + b_i^{(1)}\right)\right] + b^{(2)}\right)\end{cases} \tag{17}$$

It can be seen that equation (16) contains the first derivative and the second derivative of $\hat{g}(\tilde{X}(t))$ with respect to a, and the first derivative is

$$\begin{cases}\dfrac{\partial \hat{g}(\tilde{X}(t))}{\partial a_1} = \left(f^{(2)}(\cdot)\right)' \sum_{i=1}^{r}\left\{W_i^{(3)}\left(f_i^{(1)}(\cdot)\right)'\left[W_i^{(1)}(-s_1(t))\right]\right\}, \\ \vdots \\ \dfrac{\partial \hat{g}(\tilde{X}(t))}{\partial a_k} = \left(f^{(2)}(\cdot)\right)' \sum_{i=1}^{r}\left\{W_i^{(3)}\left(f_i^{(1)}(\cdot)\right)'\left[W_i^{(1)}(-s_k(t))\right]\right\}.\end{cases} \tag{18}$$

The component form of the first derivative in equation (18) can be written as the following vector form:

$$\frac{\partial \hat{g}(\tilde{X}(t))}{\partial a} = \left(f^{(2)}(\cdot)\right)' \sum_{i=1}^{r}\left\{W_i^{(3)}\left(f_i^{(1)}(\cdot)\right)'\left[W_i^{(1)}(-S(t))\right]\right\}. \tag{19}$$

The second derivative of $\hat{g}(\tilde{X}(t))$ with respect to a is

$$\begin{aligned}\frac{\partial^2 \hat{g}(\tilde{X}(t))}{\partial a_j^2} = {} & \left(f^{(2)}(\cdot)\right)'' \left(\sum_{i=1}^{r}\left\{W_i^{(3)}\left(f_i^{(1)}(\cdot)\right)'\left[W_i^{(1)}\left(-s_j(t)\right)\right]\right\}\right)^2 \\ & + \left(f^{(2)}(\cdot)\right)' \sum_{i=1}^{r}\left\{W_i^{(3)}\left(f_i^{(1)}(\cdot)\right)''\left[W_i^{(1)}\left(-s_j(t)\right)\right]^2\right\}, \quad j = 1, \cdots, k.\end{aligned} \tag{20}$$

Similarly, the component form of the second derivative in equation (20) can be written as the following vector form:

$$\frac{\partial^2 \hat{g}(\tilde{X}(t))}{\partial a^2} = \left(f^{(2)}(\cdot)\right)'' \sum_{i=1}^{r} \left\{ \left(\sum_{i=1}^{r} \left[W_i^{(3)} \left(f_i^{(1)}(\cdot) \right)' W_i^{(1)}(-S(t)) \right] \right)^{\mathrm{T}} W_i^{(3)} \left(f_i^{(1)}(\cdot) \right)' W_i^{(1)}(-S(t)) \right\}$$
$$+ \left(f^{(2)}(\cdot)\right)' \sum_{i=1}^{r} \left\{ \left(W_i^{(3)} \left(f_i^{(1)}(\cdot) \right)'' W_i^{(1)}(-S(t)) \right)^{\mathrm{T}} W_i^{(1)}(-S(t)) \right\}, \tag{21}$$

where $\mathbf{S}(t)$ is expressed as

$$\begin{aligned} S(t) &= \left((s(t))^{\mathrm{T}}, (s(t-\tau))^{\mathrm{T}}, (s(t-2\tau))^{\mathrm{T}}, \cdots, (s(t-(m-1)\tau))^{\mathrm{T}} \right)^{\mathrm{T}} \\ &= \left(s_1(t), s_2(t), \cdots, s_k(t) \right) \\ &= \begin{pmatrix} s_1(t) & s_2(t) & \cdots & s_k(t) \\ s_1(t-\tau) & s_2(t-\tau) & \cdots & s_k(t-\tau) \\ \vdots & \vdots & & \vdots \\ s_1(t-(m-1)\tau) & s_2(t-(m-1)\tau) & \cdots & s_k(t-(m-1)\tau) \end{pmatrix}. \end{aligned} \tag{22}$$

4.3.2. Procedure of Algorithm. We now outline the Algorithm 1.

Remark 2. Because the mathematical expression of the Jordan neural network is very complex, the difference is used to solve the approximate first and second derivatives for simplifying the calculation when the profile least square is used for estimation. The solution process of difference is as follows:

$$\begin{aligned} R(a) &= \sum_{t=n_1}^{n-1} \left[\tilde{x}(t+1) - g(\tilde{X}(t)) \right]^2 = \sum_{t=n_1}^{n-1} \left[x(t+1) - a^T s(t+1) - g(X(t) - S(t)a) \right]^2, \\ (R(a))' &= \frac{\partial R(a)}{\partial a} = \left(\frac{\partial R(a)}{\partial a_1} \cdots \frac{\partial R(a)}{\partial a_k} \right)^{\mathrm{T}} = \frac{R(a_1) - R(a_2)}{a_1 - a_2}, \\ (R(a))'' &= \frac{\partial^2 R(a)}{\partial a^2} = \frac{(R(a_1))' - (R(a_3))'}{a_1 - a_3} = \frac{(R(a_1) - R(a_2)/a_1 - a_2) - (R(a_3) - R(a_2)/a_3 - a_2)}{a_1 - a_3}, \\ &= \begin{pmatrix} \frac{\partial^2 R(a)}{\partial a_1^2} & \cdots & \frac{\partial^2 R(a)}{\partial a_1 \partial a_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 R(a)}{\partial a_k \partial a_1} & \cdots & \frac{\partial^2 R(a)}{\partial a_k^2} \end{pmatrix}. \end{aligned} \tag{23}$$

We take 3 points (a_1, a_2, a_3) near a and compute the first derivative $(R(a))'$ and the second derivative $(R(a))''$ of the function $R(a)$, where $(R(a))'$ is a vector of the same form as a. $(R(a))''$ is a matrix with k rows and k columns.

5. Experimental Simulations

In order to verify the validity and feasibility of the proposed prediction model, three simulation experiments were carried

Input: Observation sequence $x(t)$, the initial value of $a(a^{(0)})$, precision (0.00001).
Output: $a^{(1)}$
Begin
While ($a^{(1)} - a^{(0)}$ is greater than the given precision);
Step 1: For given $a^{(0)}$, update the series $x(t)$ by equation (9).
Step 2: The phase point data set $X(t)$ is obtained by normalizing and reconstructing $x(t)$ with the embedding dimension and the time delay.
Step 3: Based on (A) in Section 4.3.1 above, the Jordan neural network was trained to use the phase point data set $X(t)$.
Step 4: $a^{(1)}$ is updated using the difference.
End while
Obtain the optimal $\mathbf{a}^{(1)}$ to achieve the accuracy.
End

Algorithm 1: Jordan neural network estimation algorithm.

out. The experimental data in this paper were generated by Lorenz dynamic system and realized by R language. The codes of experiments are uploaded to GitHub website, link is https://github.com/ling-xiu/jordan/blob/master/An application of Jordan neural network. The mean square error(MSE) is used to measure the recovery accuracy of the model:

$$\begin{aligned} \mathrm{SNR}_{\mathrm{input}} &= 10\lg\left(\frac{\sigma_{s(t)}^2}{\sigma_{c(t)}^2 + \sigma_{n(t)}^2}\right), \\ \mathrm{MSE} &= \frac{1}{n}\sum_{t=1}^{n}\left[(s(t) - \widehat{s}(t))^2\right], \end{aligned} \tag{24}$$

where

$$\begin{aligned} \sigma_{s(t)}^2 &= \frac{1}{n}\sum_{t=1}^{n}(s(t) - \overline{s}(t))^2, \\ \sigma_{c(t)}^2 &= \frac{1}{n}\sum_{t=1}^{n}(c(t) - \overline{c}(t))^2. \end{aligned} \tag{25}$$

The equation of Lorenz dynamic system is in the following form:

$$\begin{cases} x' = \sigma(y - x), \\ y' = rx - y - xz, \\ z' = xy - bz, \end{cases} \tag{26}$$

where $\sigma = 10, b = 8/3$, and $r = 28$. Let x, y, and z be the time parameters that are measured according to discretization and the discretization step size is 0.01. We consider the length of data which is used to solve numerical equation (26) by the fourth-order Runge–Kutta method is 10000. We select the first component of equation (26) as the data of chaotic background noise $x(t)$. With complex autocorrelation and the method of Cao, the embedding dimension can be determined as $m = 6$ and the delay time can be determined as $\tau = 7$.

We normalize the original data sets and there are two kinds of normalized equations. Different neural network structures correspond to different normalization equations in this paper:

$$\begin{aligned} z(t) &= \frac{x(t) - E(x(t))}{\sigma_{x(t)}}, \\ z(t) &= \frac{x(t) - (x(t))_{\min}}{(x(t))_{\max} - (x(t))_{\min}}, \end{aligned} \tag{27}$$

where $x(t)$ is the original data and $z(t)$ is the normalized data sets. $E(x(t))$ is the mean of $x(t)$ and $\sigma_{x(t)}$ is the standard deviation of $x(t)$.

Absolute error (AE) and absolute percentage error (APE) are used to measure the error, and they are

$$\begin{aligned} AE &= |\widehat{a} - a|, \\ APE &= \left|\frac{\widehat{a} - a}{a}\right| \times 100\%. \end{aligned} \tag{28}$$

$\mathrm{SNR}_{\mathrm{input}}$ defines the ratio of the variance of $s(t)$ to the variance of the sum of $c(t)$ and $n(t)$.

5.1. Example 1: Prediction of Pulse Signals with Different SNRs. Chaotic time series $c(t)$ is generated by the Lorenz dynamic system and the index t ranges from 3000 to 7000 for ensuring the chaotic nature of the data. Pulse signal $s(t)$ is generated by equation (2) and the unknown parameters T_i and a are assumed as shown in Table 1. The experimental data consists of $c(t)$, $s(t)$, and white noise sequences $n(t)$ with a mean of zero.

To demonstrate the stability of the model presented in this paper, we performed Monte Carlo simulations consisting of 50 runs on the same data and the experimental results were averaged.

From Table 1 and Figure 3, we can see that the convergence rate is very fast, and the iteration times of the 12 sets of data in the experiment can all reach the given accuracy within 20 times. Second, both the absolute error (AE) and the absolute percentage error (APE) of the amplitude of the impulse signal are very small. The mean square error (MSE) of the recovered signal is less than 10^{-5}. As can be seen from Table 2, the model proposed in this paper is very stable. For these 12 sets of data, no difference in the amplitude of the recovered pulse signal was found in either one experiment or 50 Monte Carlo simulation experiments. The

Table 1: Predicted value of a, AE, and APE with different SNRs.

	T_i	a	$\text{SNR}_{\text{input}}$	$\widehat{a}$	AE	APE	MSE
(1)	130	0.1	−59.18164	0.0969737	0.0030263	0.030263	6.9×10^{-8}
(2)	130	0.12	−57.59802	0.1161813	0.0038187	0.031823	1.1×10^{-7}
(3)	140	0.15	−55.95726	0.1527467	0.0027467	0.018311	5.3×10^{-8}
(4)	140	0.22	−52.63063	0.2215733	0.0015733	0.007151	1.7×10^{-8}
(5)	150	0.3	−50.25632	0.3005346	0.0005346	0.001782	1.9×10^{-9}
(6)	150	0.4	−47.75755	0.3982338	0.0017662	0.004415	2.0×10^{-8}
(7)	150	0.5	−45.81935	0.4988764	0.0011236	0.002247	8.2×10^{-9}
(8)	180	0.8	−42.65903	0.7941931	0.0058069	0.007259	1.9×10^{-7}
(9)	200	1.1	−40.32545	1.1067487	0.0067487	0.006135	2.2×10^{-7}
(10)	200	1.5	−37.63148	1.5022624	0.0022624	0.001508	2.4×10^{-8}
(11)	200	1.9	−35.57823	1.9066362	0.0066362	0.003493	2.1×10^{-7}
(12)	210	3.6	−30.26097	3.5552166	0.0447834	0.012440	9.0×10^{-6}

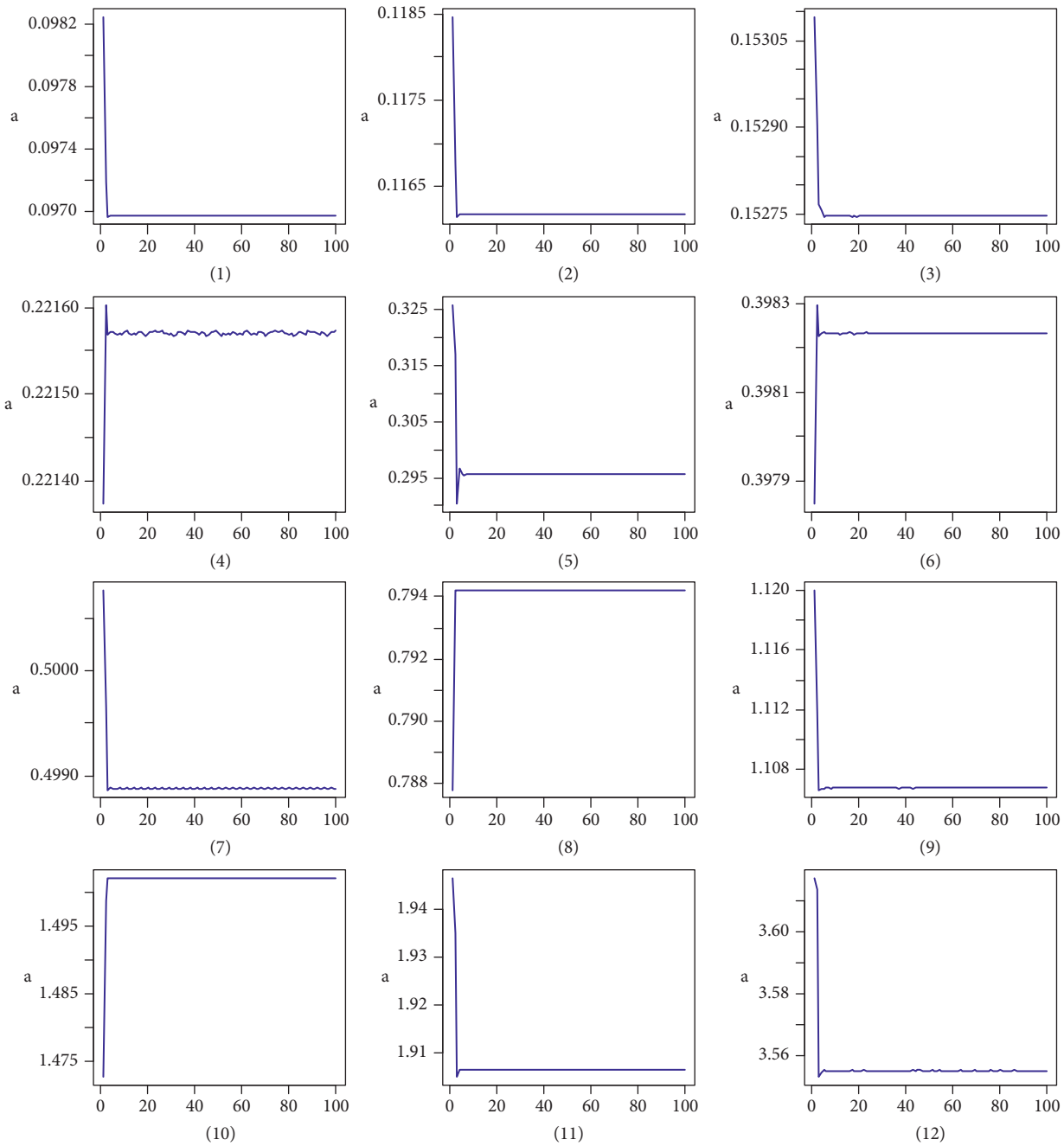

Figure 3: Iteration results of the proposed algorithm for Example 1. (1) The iteration result at $a = 0.1$. (2) The iteration result at $a = 0.12$. (3) The iteration result at $a = 0.15$. (4) The iteration result at $a = 0.22$. (5) The iteration result at $a = 0.3$. (6) The iteration result at $a = 0.4$. (7) The iteration result at $a = 0.5$. (8) The iteration result at $a = 0.8$. (9) The iteration result at $a = 1.1$. (10) The iteration result at $a = 1.5$. (11) The iteration result at $a = 1.9$. (12) The iteration result at $a = 3.6$.

Table 2: Predicted value of a and standard deviation with different SNRs.

	T_i	a	Mean of $\widehat{a}$	Standard deviation (std)
(1)	130	0.1	0.09665155	0.001731646
(2)	130	0.12	0.1161727	0.002553445
(3)	140	0.15	0.1497219	0.001991941
(4)	140	0.22	0.2191495	0.001827229
(5)	150	0.3	0.3030018	0.001105854
(6)	150	0.4	0.4028839	0.001321245
(7)	150	0.5	0.5030558	0.001121901
(8)	180	0.8	0.7990241	0.001515569
(9)	200	1.1	1.102412	0.001636155
(10)	200	1.5	1.502059	0.002687494
(11)	200	1.9	1.901625	0.004017169
(12)	210	3.6	3.563963	0.011684620

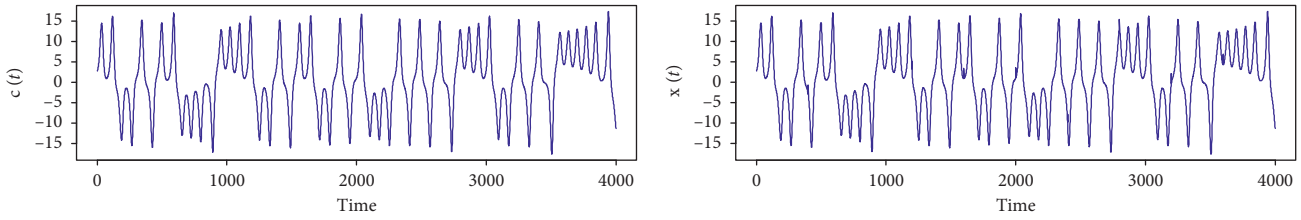

Figure 4: Data display diagram. (a) Chaotic noise $c(t)$. (b) Original data $x(t)$.

standard deviation of the results of 50 Monte Carlo simulation experiments is less than 0.02 and the standard deviation (std) increases with the increase of pulse signal amplitude.

5.2. Example 2: Prediction Comparison of Different Models. In this experiment, different models were used to fit the same set of observed data, which had an amplitude of 2.5 and a period of 400. The SNR of the observed data is −36.42872. The chaotic noise and original observation signal are shown in Figure 4. The models that fit the reconstructed original data are Jordan neural network (JNN), Elman neural network (ENN), backpropagation neural network (BPNN), support vector machine (SVM), and multilayer perceptron (MLP).

From Table 3 and Figure 5, the Jordan neural network has the smallest absolute error between the estimated amplitude and the real value, while the MLP has the largest absolute error. It can be seen from in Figures 6(a), 6(c), and 6(e) that the fitted value of the original signal by Jordan neural network is significantly closer to the real value than support vector machine (SVM) and multilayer perceptron (MLP). It can be seen from the prediction error graph of Figures 6(b), 6(d), and 6(f) that the prediction error of Jordan neural network is relatively large at the time point with pulse signal, and the error value is close to the amplitude of the pulse signal, while the time point error of the rest without pulse signal is relatively small. Support vector machine and multilayer perceptron have large prediction error.

5.3. Example 3: Jordan Neural Network Compared with Backpropagation Neural Network. $\mathrm{SNR}_{\mathrm{output}}$ is the signal-to-noise ratio after extracting the pulse signal. $\mathrm{SNR}_{\mathrm{output}}$ defines the ratio of the variance of $\widehat{s}(t)$ to the variance of residual:

$$\mathrm{SNR}_{\mathrm{output}} = 10\lg\left(\frac{\sigma^2_{\widehat{s}(t)}}{\sigma^2_{x(t)-\widehat{x}(t)}}\right). \tag{29}$$

Figure 7 takes the input SNR as the horizontal axis and the output SNR as the vertical axis. The experimental data are 12 sets of data from experiment 1, and then the backpropagation neural network estimation is carried out on the basis of experiment 1. It can be seen from Figure 7 that the output SNR increases with the increase of the input SNR, and they are proportional to each other. And with the increase of the input SNR, the increase of the output SNR slows down. It can also be seen from the figure that the output SNR of the Jordan neural network is 10–20db higher than the output SNR of the backpropagation neural network.

5.4. Discussion. This section reports the performance of the profile method combined with Jordan neural network in estimation of pulse signals against chaotic background. The purpose of the experiments was to evaluate if JNN can maintain quality in prediction performance when compared to conventional methods.

The results have shown that the method proposed in this paper not only solves the weak pulse signal in chaotic signal well, but also has good stability. We evaluate the application breadth of the proposed model by comparing the experimental data of different SNR. Table 1 shows that the

Table 3: Predicted value of a with different models.

Model	a	Mean of $\hat{a}$	AE	APE	std
JNN	2.5	2.501952	0.001952	0.0007808	0.004211562
BPNN	2.5	2.503297	0.003297	0.0013188	0.006069684
ENN	2.5	2.503410	0.003410	0.0013640	0.005308634
SVM	2.5	2.387866	0.112134	0.0448536	0.010575260
MLP	2.5	2.969629	0.469629	0.1878516	0.091993390

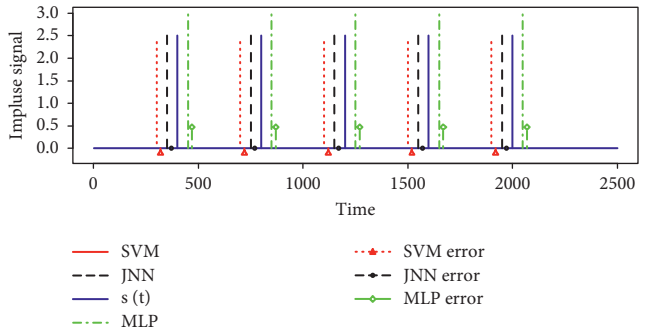

Figure 5: Comparison diagram of different models.

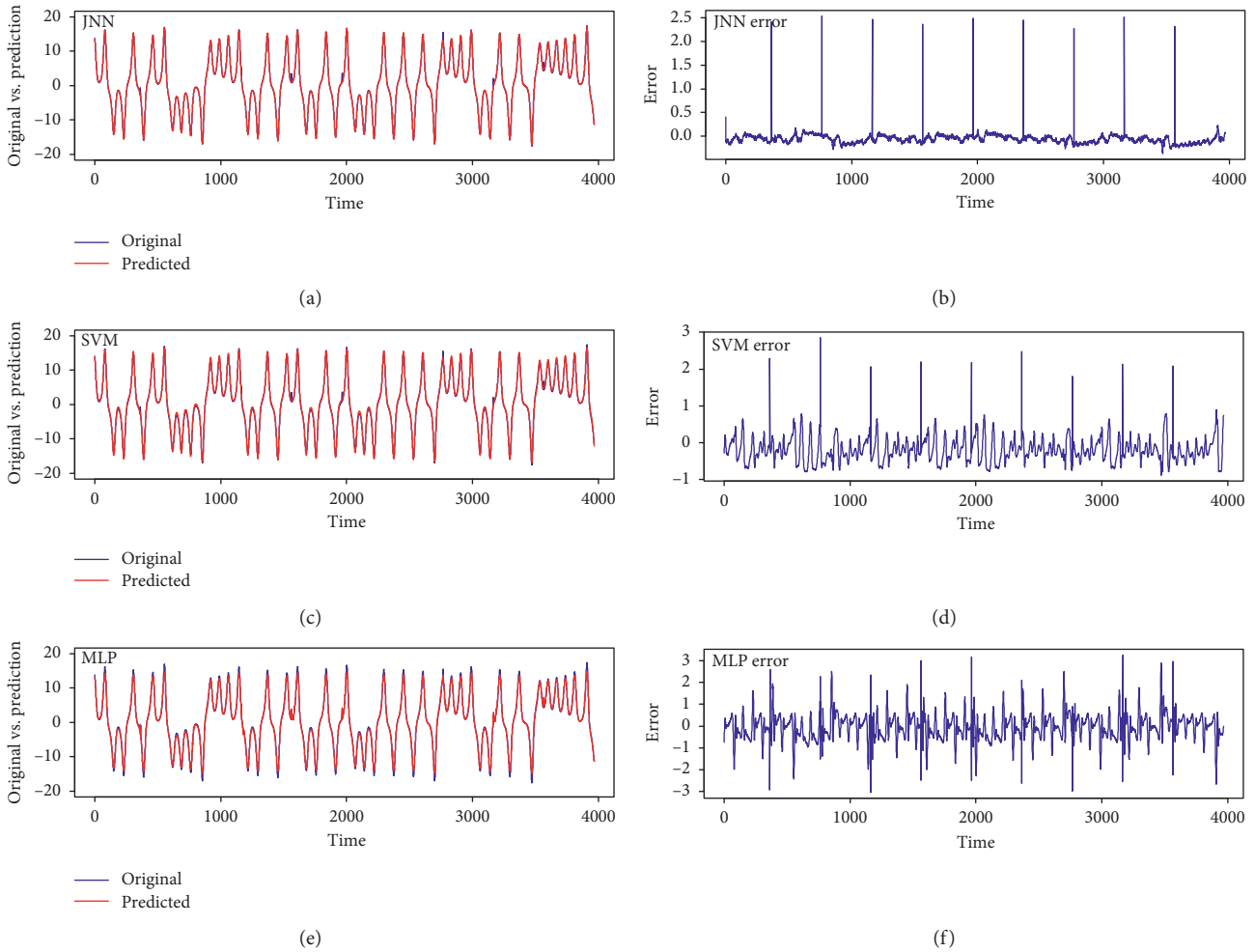

Figure 6: Comparison diagram of original signal and predicted signal and the predicted error diagram: (a) original signal vs. the result of Jordan neural network; (b) Jordan neural network prediction error; (c) original signal vs. the result of support vector machine; (d) support vector machine prediction error; (e) original signal vs. the result of multilayer perceptron; (f) multilayer perceptron prediction error.

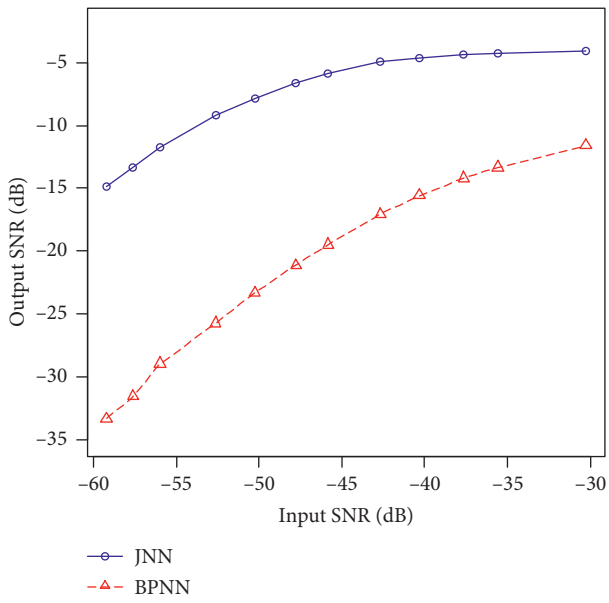

Figure 7: Output SNR vs. input SNR.

estimated SNR range of the proposed model is −60 to −30. Within this range, the model proposed in this paper can effectively estimate the pulse signal amplitude, and its absolute error (AE) can be as low as 0.00112 and as high as 0.05, and its MSE is also below 10^{-5}.

The results report the estimation of pulse signals by different neural networks under chaotic noise background, as well as the mean amplitude and standard deviation (std) of 50 test runs in Table 3. We can also see from Table 3 that the two minimum standard deviations are 0.00421 and 0.00531, which are, respectively, from the Jordan neural network and the Elman neural network. We know that both the Jordan neural network and the Elman neural network are feedback neural networks. The difference between them is that the context layer connects to a different layer. Compared with [35], we found that the Jordan neural network could better fit the time series than the Elman neural network.

A major innovation of this method is to use Jordan neural network to fit the chaotic noise background and then estimate the impulse signal in the chaotic noise by combining the profile method. Compared with other neural networks, this method is more accurate and the overall training time is shorter.

6. Conclusions

In this paper, we are interested in weak pulse signal in chaotic background. Based on the short-term predictability and Takens theorem, we provide an algorithm for directly estimating the problems; that is, the Jordan neural network is used to fit the chaotic time series, and the one-step prediction error was obtained. Then, starting from the error, the single-point jump signal model was connected, and the amplitude of the pulse signal was estimated by the profile least square method, so as to achieve the prediction effect of the pulse signal under the chaotic background. The following conclusions can be drawn from the experimental results: the model proposed in this paper can predict the weak pulse signal in the chaotic background, and it can be seen from the results of experiment 1 that the prediction accuracy is high. It can be seen from the results of experiment 2 that the Jordan neural network is significantly better than other feedforward neural networks in fitting the nonlinear dynamic system, and the absolute percentage error of the Jordan neural network is the smallest. In future work, we can try to improve in two aspects, so that it has the best generalization performance in most cases, and apply the method to solve the relevant practical problems. Firstly, we can try to estimate the amplitude of the impulse signal without knowing the period of the impulse signal. Secondly, the Jordan neural network can be trained by other optimization methods, and the Jordan neural network can be improved.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Fundamental and Advanced Research Project of CQ CSTC of China (Grant no. cstc2018jcyjAX0464).

References

[1] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963.

[2] T. Jin and H. Zhang, "Statistical approach to weak signal detection and estimation using Duffing chaotic oscillators," *Science China Information Sciences*, vol. 54, no. 11, pp. 2324–2337, 2011.

[3] L. Su, H. Sun, J. Wang, and L. Yang, "Detection and estimation of weak pulse signal in chaotisc background noise," *Acta Physica Sinica*, vol. 66, no. 9, Article ID 090503, 2017.

[4] L. Su and C. Li, "Extracting narrow-band signal from a chaotic background with LLVCR," *Wireless Personal Communications*, vol. 96, no. 2, pp. 1907–1927, 2017.

[5] C. Li and L. Su, "Extracting harmonic signal from a chaotic background with local linear model," *Mechanical Systems and Signal Processing*, vol. 84, pp. 499–515, 2017.

[6] X. Wang, "Multivariate chaotic time series prediction using multiple kernel extreme learning machine," *Acta Physica Sinica*, vol. 64, no. 7, Article ID 070504, 2015.

[7] X. Wu and Y. Wang, "Extended and Unscented Kalman filtering based feedforward neural networks for time series prediction," *Applied Mathematical Modelling*, vol. 36, no. 3, pp. 1123–1131, 2012.

[8] L. Su, H. Sun, and C. Li, "LL-P-KF hybrid algorithm for detecting and recovering sinusoidal signal in strong chaotic noise," *Acta Electronica Sinica*, vol. 45, no. 4, pp. 837–843, 2017.

[9] L. Su, Li Deng, W. Zhu, and S. Zhao, "Detection and extraction of weak pulse signals in chaotic noise with PTAR and DLTAR models," *Mathematical Problems in Engineering*, vol. 2019, Article ID 4842102, 12 pages, 2019.

[10] X. Jiang, "Prediction of hydrologic chaotic time series using volterra-NLMS adaptive filter," *Journal of Applied Statistics and Management*, vol. 34, no. 3, pp. 434–441, 2015.

[11] M. Han, Y. Liu, Z. Shi, and Mu Xiang, "The study of chaotic noise reduction method with improved local projection," *Journal of System Simulation*, vol. 19, no. 2, pp. 364–368, 2007.

[12] Y. Wang, C. Shi, G. Qian, and W. Wang, "Prediction of chaotic time series based on the fractional-order maximum correntropy criterion algorithm," *Acta Physica Sinica*, vol. 67, no. 1, Article ID 018401, 2018.

[13] H. Lu, D. Li, and H. Sun, "Prediction for chaotic time series of optimized BP neural network based on PSO," *Computer Engineering and Applications*, vol. 51, no. 2, pp. 224–229, 2015.

[14] L. Yin, Y. He, X. Dong, and Z. Lu, "Adaptive chaotic prediction algorithm of RBF neural network filtering model based on phase space reconstruction," *Journal of Computers*, vol. 8, no. 6, pp. 1449–1455, 2013.

[15] S. Li, Y. Zhu, C. Xu, and Z. Zhou, "Study of personal credit evaluation method based on PSO-RBF neural network model," *American Journal of Industrial and Business Management*, vol. 03, no. 4, pp. 429–434, 2013.

[16] Y.-Y. Fu, C.-J. Wu, J.-T. Jeng, and C.-N. Ko, "ARFNNs with SVR for prediction of chaotic time series with outliers," *Expert Systems with Applications*, vol. 37, no. 6, pp. 4441–4451, 2010.

[17] L. Han, L. Ding, and H.-P. Ren, "Chaos control based on least square support vector machines," *Acta Physica Sinica*, vol. 54, no. 9, pp. 4019–4024, 2005.

[18] X. Zhou, Li Liao, M. Zhang, and Q. Sheng, "Speech recognition based on fuzzy neural network and chaotic differential evolution algorithm," *Journal of Information and Computational Science*, vol. 12, no. 14, pp. 5451–5458, 2015.

[19] R. Chandra and M. Zhang, "Cooperative coevolution of elman recurrent neural networks for chaotic time series prediction," *Neurocomputing*, vol. 86, pp. 116–123, 2012.

[20] R. Chandra, "Competition and collaboration in cooperative coevolution of elman recurrent neural networks for time-series prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 3123–3136, 2015.

[21] Z. Shi and M. Han, "Support vector echo-state machine for chaotic time-series prediction," *IEEE Transactions on Neural Networks*, vol. 18, no. 2, pp. 359–372, 2007.

[22] R. Chandra, "Memetic cooperative coevolution of Elman recurrent neural networks," *Soft Computing*, vol. 18, no. 8, pp. 1549–1559, 2013.

[23] M. I. Jordan, "Serial order: a parallel distributed processing approach," Tech. Rep. No. 8604, University of California, Institute for Cognitive Science, San Diego, CA, USA, 1997.

[24] P. Peng, W. Zhang, Yi Zhang, Y. Xu, H. Wang, and H. Zhang, "Cost sensitive active learning using bidirectional gated recurrent neural networks for imbalanced fault diagnosis," *Neurocomputing*, vol. 407, pp. 232–245, 2020.

[25] S. Jeon and J. Moon, "Malware-detection method with a convolutional recurrent neural network using opcode sequences," *Information Sciences*, vol. 535, pp. 1–15, 2020.

[26] R. Gao, Y. Huo, S. Bao et al., "Multi-path X-D recurrent neural networks for collaborative image classification," *Neurocomputing*, vol. 397, pp. 48–59, 2020.

[27] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *International Journal of Forecasting*, vol. 36, no. 1, pp. 75–85, 2020.

[28] Y. Liu, C. Gong, L. Yang, and Y. Chen, "DSTP-RNN: a dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction," *Expert Systems With Applications*, vol. 143, Article ID 113082, 2019.

[29] W. Wu, S. An, P. Guan, D. Huang, and B. Zhou, "Time series analysis of human brucellosis in mainland China by using Elman and Jordan recurrent neural networks," *BMC Infectious Diseases*, vol. 19, no. 1, 2019.

[30] Z. Wang, G. Horng, and T. Hsu, "Heart sound signal recovery based on time series signal prediction using a recurrent neural network in the long short-term memory model," *The Journal of Supercomputing*, vol. 1, 2019.

[31] Y. Zhang, Y. Wang, and G. Luo, "A new optimization algorithm for non-stationary time series prediction based on recurrent neural networks," *Future Generation Computer Systems*, vol. 102, pp. 738–745, 2020.

[32] F. Takens, "Detecting strange attractors in turbulence," *Dynamical Systems and Turbulence, Warwick*, pp. 366–381, Springer, Berlin, Germany, 1980.

[33] L. Cao, "Practical method for determining the minimum embedding dimension of a scalar time series," *Physica D: Nonlinear Phenomena*, vol. 110, no. 1-2, pp. 43–50, 1997.

[34] J. Fan and Q. Yao, *Nonlinear Time Series: Nonparametric and Parametric Methods*, pp. 337–348, Springer-Verlag, Heidelberg, Germany, 2003.

[35] N. Mohana Sundaram and N. Sivanandam, "A hybrid elman neural network predictor for time series prediction," *International Journal of Engineering & Technology*, vol. 7, no. 2, pp. 159–163, 2018.