TIME SERIES PREDICTION USING CONVOLUTION SUM DISCRETE PROCESS NEURAL NETWORK

*Ding Gang, Lei Da, Zhong Shisheng**

Abstract: A convolution sum discrete process neural network (CSDPNN) is proposed. CSDPNN utilizes discrete samples as inputs directly and employs convolution sum to simulate the process inputs so as to deal with the time accumulation existing in many time series. Without the procedures of fitting the discrete samples into continuous functions to generate inputs and then to expand the input functions by basis functions, CSDPNN is better understandable and is with less precision reduction compared with process neural network (PNN) with function inputs. The approximation capacity of CSDPNN is analyzed in this paper, and it proved that CSDPNN can approximate PNN and has approximation capacity not worse than traditional artificial neural network (ANN). Finally, CSDPNN, PNN and ANN are utilized to predict the Logistic chaos time series and the iron concentration in the lubrication oil of aircraft engine, and the application test results indicate that CSDPNN performs better than PNN and ANN given the same conditions.

Key words: *Process neural network, functional approximation, time series prediction, aeroengine health condition monitoring,*

Received: October 18, 2013 **DOI:** 10.14311/NNW.2014.24.025
Revised and accepted: August 20, 2014

1. Introduction

Time accumulation effect widely exists in practical systems, which leads to the fact that the systems' responses not only depend on the instantaneous inputs but also rely on the inputs before the instantaneous moment. In other words, responses of such systems are time-depending processes. HE proposed the process neural network (PNN) which has the ability to deal with the time varying process so as to handle such problems [6]. PNN has a similar structure with the traditional artificial neural network (ANN), and is composed of weights, aggregate and activation, but the aggregate operation unit of PNN contains spatial aggregate operation and time aggregate operation, where the time aggregation operator deals with the time varying process. Taking the advantages of processing time accumulation effects,

*Ding Gang – Corresponding Author, Lei Da, Zhong Shisheng, School of Mechatronics Engineering, Harbin Institute of Technology, Harbin 150001, P.R.China Phone: (+86)139 4609 2735, E-mail: dingganghit@163.com

PNN can often get higher precision in time series prediction compared with NN, and is widely used in such areas [2], [3], [8], [9], [10].

To simulate the continuous inputs and time accumulation process, PNN requires continuous function inputs and integral operations on the input functions. However, what we have are always discrete samples in practical applications. The current solution to such problem is to fit the discrete samples to generate continuous function inputs firstly, then to expand the input functions and the weight functions based on orthogonal basis functions to simplify the burdensome integral operations [5]. This solution has some disadvantages: (1) the discrete samples may not be fitted into analytic functions, and the fitting may cause some precision loss, (2) while the fitting order is high, there may be Runge oscillation which leads to distortion, (3) there is no theoretical directions for choosing the basis functions while decomposing the input functions and the weight functions, (4) the function decomposing process may cause precision loss too.

In this paper, a convolution sum discrete process neural network (CSDPNN) is proposed. This new neural network model utilizes the discrete data as input directly, and adopts the convolution sum to deal with the time varying process based on signal processing theory. Without procedures of function fitting and function decomposing, there is less precision loss which can lead to higher precision while applying the new network to predict time series.

The paper is organized as follows. Section 2 describes basic theory of the CSDPNN. Section 3 develops a learning algorithm for CSDPNN. Section 4 analyzes the approximation capacity of CSDPNN. In section 5, CSDPNN is utilized to predict the Logistic chaos time series and the iron concentration of the aircraft engine lubricating oil, and the prediction results are compared with results generated by PNN and ANN. Discussions and conclusions are given in section 6.

2. The Convolution Sum Discrete Process Neural Network

2.1 Convolution Sum Discrete Process Neuron

According to the theory of time-invariant system, if the system input is discrete series $\mathbf{x}(n)$, then the response of the system $\mathbf{y}(n)$ is given as

$$\mathbf{y}(n) = \mathbf{x}(n) * \mathbf{h}(n) = \sum_{k=-\infty}^{\infty} \mathbf{x}(k)\mathbf{h}(n-k) \tag{1}$$

where $\mathbf{h}[n]$ is the system response to the unit pulse series. Equation (1) means that the response of linear system on time n equals the summation of the system responses to the whole inputs on time n. If k in Equation (1) begins with 0, and the input series $\mathbf{x}[n]$ is with limited length of n which can be seemed as a continuous input process, then the system response on time n can be seemed as the summation of the system responses between time 0 and time n. That's to say, the system response depends on the continuous input of a whole period not just the instantaneous moment n. From the physical meaning of the time-invariant

system, we can conclude that the convolution sum can be utilized to stimulate the continuous input process and its corresponding response of the biological neurons.

Based on the aforementioned analysis, the convolution sum discrete process neuron was developed in this paper. As depicted in Fig. 1, it is composed of three sections: inputs, an activation unit and an output unit.

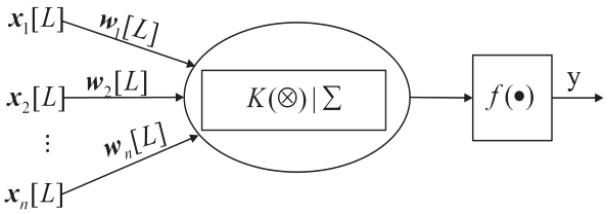

Fig. 1 *Sketch diagram of CSDPNN neuron model.*

The inputs of the CSDPNN neuron are vectors with length of L, which are composed of discrete sampling points in time interval $[0, T]$, and the corresponding weights are also vectors with length of L. $\sum$ denotes weighted sums, which implements the spatial aggregate operation, and $K(\otimes)$ denotes convolution sum, which implements the time aggregate operation. For discrete neuron showed in Fig. 1, we have

$$K(\otimes) = \sum (\mathbf{x}_i * \mathbf{w}_i) = \sum (\mathbf{w}_i * \mathbf{x}_i) \tag{2}$$

Where $*$ denotes convolution operation. According to the calculation rule of convolution sum, we have

$$K_i(\otimes)(n) = \mathbf{x}_i * \mathbf{w}_i = \sum_{k=-\infty}^{\infty} x_i(k) w_i(n-k) \tag{3}$$

Since the length of vector $\mathbf{x}_i$is L, the values of variable k in Equation (3) can be $1, 2, \cdots, L$, and when the value of n in equation (3) is L, we have

$$K_i(\otimes)(L) = \mathbf{x}_i * \mathbf{w}_i = \sum_{k=1}^{L} x_i(k) w_i(L-k) \tag{4}$$

Let $\omega_j(k) = w_i(L-k)$, then Equation (4) can be rewritten as

$$K_i(\otimes)(L) = \mathbf{x}_i * \omega_i = \omega_i * \mathbf{x}_i = \sum_{k=1}^{L} x_i(k) \omega_i(k) \tag{5}$$

Now, every element of vector $\mathbf{x}_i$ attributes to the value of $K_i(\otimes)(L)$. So, $K_i(\otimes)(L)$ is selected as the time aggregate operation results of the CSDPNN neuron, and denote the result as $K_i(\otimes)_L$. Then, the mapping relation of the inputs and outputs can be described as

$$y = f\left(\sum K(\otimes)_L + \theta\right) = f\left(\sum_{i=1}^{n} \mathbf{x}_i * \omega_i + \theta\right) = f\left(\sum_{i=1}^{n} \sum_{k=1}^{L} x_i(k) \omega_i(k) + \theta\right) \tag{6}$$

where $f(\bullet)$ is the activation function and θ is the activation threshold. And the Sigmoid function is utilized as the activation function in this paper.

2.2 The CSDPNN Model

The CSDPNN can be composed by several CSDPNN neurons with a special topological structure. The CSDPNN utilized in this paper is comprised of three layers. The topological structure of the CSDPNN is $n-m-1$, which is depicted in Fig. 2.

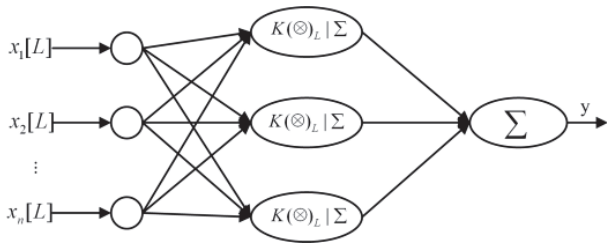

Fig. 2 *The topological structure of the CSDPNN.*

There are n input units, which are vectors with length of L. The hidden layer has m CSDPNN neurons, and the output layer has 1 ANN neuron. Supposing that the activation function of the output layer is a linear function, thus the output of the PPNN model can be expressed as

$$y = \sum_{k=1}^{m} v_k f\left(\sum_{j=1}^{m}\sum_{i}^{n} \omega_{ji} * \mathbf{x}_i + \theta_1^j\right) + \theta_2 = \sum_{k=1}^{m} v_k f\left(\sum_{j=1}^{m}\sum_{i}^{n}\sum_{l=1}^{L} \omega_{ji}(l) x_i(l) + \theta_1^j\right) + \theta_2 \tag{7}$$

Where$\omega_{\mathbf{ji}}$ is the connection weight vector between the j-th CSDPNN neuron in the hidden layer and the i-th unit in the input layer. $\mathbf{x}_i$ is the i-th unit in the input layer, θ_1^j is the threshold of the j-th CSDPNN neuron in the hidden layer, $v_k$is the connection weight between the k-th neuron in the hidden layer and the output layer, andθ_2 is the threshold of the unit in the output layer.

3. Learning Algorithm

3.1 Learning Algorithm Derivation

Levenberg-Marquardt (LM) is a widely used learning algorithm for neural networks [1], [4], [7]. Given suitable parameters, the LM algorithm can simulate the Gradient Descent algorithm and the Gauss-Newton algorithm, so that it can ensure the convergence speed while avoiding trapping in local minimum. This paper takes advantages of the LM algorithm to develop learning algorithm for CSDPNN.

Given S learning samples $\{\mathbf{x}_1, \cdots, \mathbf{x}_s; d_s\}_{s=1}^{S}$, where $\mathbf{x}_s$ the s-th input vector with length of L, and $d_s$is the corresponding target. Suppose $y_s$is the actual

corresponding output of the s-th input of the CSDPNN, then the sum squared error(SSE) of the CSDPNN output and the actual output can be defined as

$$SSE = \sum_{s=1}^{S} [d_s - y_s]^2 = \sum_{s=1}^{S} [d_s - \left(\sum_{k=1}^{m} v_k f(\sum_{j=1}^{m} \sum_{i}^{n} \sum_{l=1}^{L} \omega_{ji}(l) x_{is}(l) + \theta_1^j) + \theta_2 \right)]^2 \tag{8}$$

For the simplicity of analysis, suppose that $\mathbf{E}^T = [e_1, e_2, \cdots, e_S]$, where $e_s = d_s - y_s$, and $\mathbf{E}^T$ denotes the transposition of $\mathbf{E}$. Suppose that $\mathbf{W}^T = [\omega_{11}(1), \cdots, \omega_{11}(L), \cdots, \omega_{mn}(1), \cdots, \omega_{mn}(L), \theta_1^1, \cdots, \theta_1^m, v_1, \cdots, v_m, \theta_2]$, where $\mathbf{W}^T$ is the transposition of $\mathbf{W}$. Obviously, $\mathbf{W}$ is the parameter to be tuned.

According to the LM algorithm, during the training process, the iteration rule of $\mathbf{W}$ can be defined as

$$\begin{cases} \Delta \mathbf{W}(q) = -[\mathbf{J}^T(\mathbf{W}(q))\mathbf{J}(\mathbf{W}(q)) + \mu(q)\mathbf{I}]^{-1}\mathbf{J}^T(\mathbf{W}(q))E(\mathbf{W}(q)) \\ \mathbf{W}(q+1) = \mathbf{W}(q) + \Delta \mathbf{W}(q) \end{cases} \tag{9}$$

Where, q is the iteration number, $\mathbf{I}$ is the unit matrix, and μ is the learning rate. At the beginning of the training process, μ is given a small value. If SSE is not reduced in the iteration, then set μ with a new value according to $\mu = \mu \cdot \lambda (\lambda > 1)$ and repeat the iteration until SSE is reduced, and if SSE reduced in the iteration, then the value of μ can be set according to $\mu = \mu / \lambda$. This means that SSE can be reduced every iteration of the LM learning algorithm. $\mathbf{J}(\mathbf{W})$ is the Jacobi matrix of $\mathbf{W}$ with dimensions of $S \times L$, and it can be written as

$$\mathbf{J}(\mathbf{W}) = \begin{bmatrix} \frac{\partial e_1}{\partial \omega_{11}(1)}, \cdots, \frac{\partial e_1}{\partial \omega_{11}(L)}, \cdots, \frac{\partial e_1}{\partial \omega_{mn}(1)}, \cdots, \frac{\partial e_1}{\partial \omega_{mn}(L)}, \\ \frac{\partial e_2}{\partial \omega_{11}(1)}, \cdots, \frac{\partial e_2}{\partial \omega_{11}(L)}, \cdots, \frac{\partial e_2}{\partial \omega_{mn}(1)}, \cdots, \frac{\partial e_2}{\partial \omega_{mn}(L)}, \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ \frac{\partial e_S}{\partial \omega_{11}(1)}, \cdots, \frac{\partial e_S}{\partial \omega_{11}(L)}, \cdots, \frac{\partial e_S}{\partial \omega_{mn}(1)}, \cdots, \frac{\partial e_S}{\partial \omega_{mn}(L)}, \\ \frac{\partial e_1}{\partial \theta_1^1}, \cdots, \frac{\partial e_1}{\partial \theta_1^m}, \frac{\partial e_1}{\partial v_1}, \cdots, \frac{\partial e_1}{\partial v_m}, \frac{\partial e_1}{\partial \theta_2} \\ \frac{\partial e_2}{\partial \theta_1^1}, \cdots, \frac{\partial e_2}{\partial \theta_1^m}, \frac{\partial e_2}{\partial v_1}, \cdots, \frac{\partial e_2}{\partial v_m}, \frac{\partial e_2}{\partial \theta_2} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ \frac{\partial e_S}{\partial \theta_1^1}, \cdots, \frac{\partial e_S}{\partial \theta_1^m}, \frac{\partial e_S}{\partial v_1}, \cdots, \frac{\partial e_S}{\partial v_m}, \frac{\partial e_S}{\partial \theta_2} \end{bmatrix} \tag{10}$$

Let $U_k = \sum_{j=1}^{m} \sum_{i}^{n} \sum_{l=1}^{L} \omega_{ji}(l) x_{is}(l) + \theta_1^j$, then the elements of the Jacobi matrix $\mathbf{J}(\mathbf{W})$ can be calculated as

$$\begin{cases} \frac{\partial e_s}{\partial \omega_{ji}(l)} = -v_j f'(U_k) x_{is}(l) \\ \frac{\partial e_s}{\partial \theta_1^j} = -v_k f'(U_k) \\ \frac{\partial e_s}{\partial v_k} = -f(U_k) \\ \frac{\partial e_s}{\partial \theta_2} = -1 \end{cases} \tag{11}$$

while the Jacobi matrix $\mathbf{J}(\mathbf{W})$can be calculated based on Equation (11), the CSDPNN can be trained with parameters updating according to Equation (9).

3.2 Learning Algorithm Description

The LM learning algorithm for CSDPNN can be described as follows

Step1 Determine the structure of the CSDPNN.

Step2 Determine the learning error precision ε, the max learning iteration number M, and initialize the training parameters $\omega_{ji}, v_k, \theta_\mathbf{1}, \theta_2$ and the iteration number q.

Step3 Input all the training samples into the network, and calculate the actual outputs based on the Equation (7), then the training error with $e_s = d_s - y_s$, finally *SSE* according to Equation (8).

Step4 Calculate the Jacobi matrix based on Equation (10) and Equation (11).

Step5 Calculate *SSE* based on Equation (8). If the new value of *SSE* is bigger than the value of *SSE* calculated in step4, then set $\mu = \mu \cdot \lambda$, and repeat Step5, else if the new value of SSE is small than the value of *SSE* calculated in step4, then set $\mu = \mu/\lambda$,$q = q + 1$ turn to step6.

Step6 If *SSE* is less than ε or $q > M$, turn to step7, else turn to step5.

Step7 Output the learning results and terminate the learning process.

4. Approximation Capacity Analysis

In this section, the approximation capacity is described by two theorems as follows, also with their proofs.

Theorem 1. CSDPNN is a special case of PNN and CSDPNN has the ability of approximating the PNN with continuous function inputs.

Proof. The topology structure of PNN with continuous function inputs which has n inputs and m neurons in the hidden layer and 1 neuron in the output layer can be described as $n - m - 1$, and the map relationship between the input layer and the output layer can be described as

$$y = \sum_{j=1}^{m} v_j f\left(\int_0^T \sum_{j=1}^{m}\sum_{i=1}^{n} w_{ji}(t)x_i(t)dt - \theta_j^1\right) - \theta_2 \tag{12}$$

where y denotes the output of the network, $x_i(t)(x_i(t) \in C[0,T])$ is the i-th input unit in the input layer, and $w_{ji}(t)(w_{ji}(t) \in C[0,T])$ is the connection weight function between the $j-$th unit of the hidden layer and the $i-$th unit in the input layer.

Let $u_j = \int_0^T \sum_{i=1}^n w_{ji}(t)x_i(t)dt$, then according to the definition of integration, we can conclude than the integration on the defined interval $[0, T]$ can be approximated by weighted sums as

$$\left| \int_0^T \sum_{j=1}^m \sum_{i=1}^n w_{ji}(t)x_i(t)dt - \sum_{k=1}^L \sum_{j=1}^m \sum_{i=1}^n \frac{T}{L} w_{ji}(k)x_i(k) \right|^2 < \varepsilon \tag{13}$$

Where $x_i(k)$ and $w_{ji}(k)$ are the value of the i-th input function and the weight function between the j-th unit in the hidden layer and the i-th unit in the input layer on time k respectively, L is the total number of the sub-intervals generated by equally dividing the interval $[0, T]$, and $\varepsilon$is a arbitrarily small positive number. Because $w_{ji}(k)$ is the weight to be tuned, we can define a new weight as $\omega_{ji}(k) = \frac{T}{L} w_{ji}(k)$, then Equation (13) can be rewritten as

$$u_j = \sum_{k=1}^L \sum_{i=1}^n \omega_{ji}(k)x_i(k) = \sum_{i=1}^n \sum_{k=1}^L \omega_{ji}(k)x_i(k) \tag{14}$$

So, the Equation (12) has the following transformation

$$y = \sum_{j=1}^m v_j f\left(\sum_{j=1}^m \sum_{i=1}^n \sum_{k=1}^L \omega_{ji}(k)x_i(k) - \theta_j^1 \right) - \theta_2 \tag{15}$$

It has similar formation as the Equation (7) which the mapping relationship between the inputs and the output of CSDPNN. So we can conclude that CSDPNN is a special case of PNN with continuous function inputs, and CSDPNN can approximate PNN with continuous function inputs within arbitrarily precision by giving the input vector with enough length.

Theorem 2. Traditional artificial neural network is a special case of CSDPNN.

Proof. Let$L = 1$, then the mapping relationship between the inputs and the output of CSDPNN can be written as

$$y = \sum_{j=1}^m v_j f\left(\sum_{j=1}^m \sum_{i=1}^n \omega_{ji}x_i - \theta_j^1 \right) - \theta_2 \tag{16}$$

Obviously, Equation (16) is the mapping relationship between the inputs and the outputs of ANN.

5. Application Test

5.1 Logistic time series prediction

Logistic time series is a kind of chaos series, which is widely used to test the performance of predicting algorithms. In this section, CSDPNN proposed in this paper is utilized to predict the Logistic time series.

Logistic time series can be generated according to the following equation

$$x(n+1) = kx(n)\,(1 - x(n)) \quad 0 < k \leq 4 \tag{17}$$

when $k \approx 3.5$, the logistic system begins to generate chaos series. Let $k = 3.6$, and the initial value of x which is $x(0) = 0.4$, then a series with length of 1206 based on Equation(17) can be generated, and the later 206 data of the series are selected as test samples, which can be denoted as $\{x_k\}_{k=1}^{206}$. $(x_i, x_{i+1}, \cdots, x_{i+5})$ can be selected to form the input vector IV_i, $i = 1, \cdots, 200$, and x_{i+6} to be the corresponding output. Then, we can get 200 couples of samples which can be denoted as $\{IV_i, x_{i+6}\}_{i=1}^{200}$, and the former 100 couples are selected to train the CSDPNN and the left 100 couples are selected to test out model. The topological structure of the used CSDPNN is $1 - 10 - 1$, the error goal is set to 0.001, and the max iteration number is set to 1000. The prediction results are depicted in Fig. 3, it indicates that CSDPNN has very high precision on predicting the Logistic chaos time series in this paradigm.

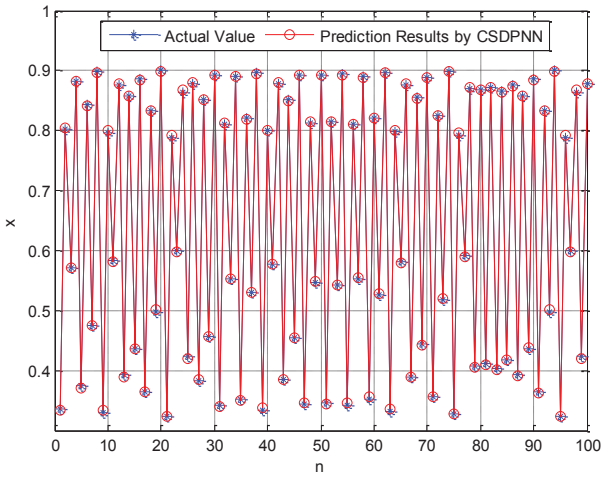

Fig. 3 *Logistic Time Series Prediction Results by CSDPNN.*

PNN and multilayered ANN are utilized for comparison of performance in this section. For PNN, We choose $(x_i, x_{i+1}, \cdots, x_{i+5})$ to be an input function IF_i, $i = 1, \cdots, 200$, and x_{i+6} to be the corresponding output. Then, we can get 200 couples of samples which can be denoted as $\{IF_i, x_{i+6}\}_{i=1}^{200}$. As the same with the CSDPNN, the former 100 couples are used to train the PNN and the left 100 couples are used to test the PNN. Also, the parameters for PNN are set as the same with CSDPNN. For ANN, the network structure is different from CSDPNN and PNN, since ANN cannot use functions of vectors as inputs, and topology structure of ANN used in this section is 6-10-1.

The prediction errors of the results are presented in Tab. I. It can be concluded from Tab. I that CSDPNN has the best performance in this paradigm, since CSDPNN has smaller mean absolute error (MAE) and smaller max absolute error than its counterparts.

	Absolute error	
	Max	MAE
CSDPNN	0.011	**0.002**
PNN	0.012	0.004
ANN	0.015	0.005

Tab. I *Prediction errors of logistic time series.*

5.2 Fe concentration time series prediction

Aircraft engine is a complicated nonlinear system, which is always working under extreme conditions such as high temperature, high pressure and high speed leading to the result that the performance of its components and subsystems will degrade with time. So condition monitoring is essential in terms of flight safety and also for reduction of the preventive maintenance cost. Lubrication oil monitoring is one of the most important aspects of condition monitoring for aero-engine. Analysis of the lubricating oil taken from the aircraft engine gives an indication of its suitability for continued use and provides important information about the health condition of the lubricated components within the aircraft engine. In this section, the CSDPNN is utilized to predict the iron (Fe) concentration of the lubricating oil in the aircraft engine condition monitoring to highlight the approximation capability of the CSDPNN.

The Fe concentration time series is from an airline company in China, which is depicted in Fig. 4. The sampling interval of the data used in this paper is about 143.5 hours and the sampling interval is not equal. Thus, the time series need to be preprocessed before the prediction. So, the cubic spline interpolation method is utilized to generate a new time series. And after the interpolation, we get a Fe concentration time series with 141 discrete points denoted as $\{Fe_j\}_{j=1}^{141}$.

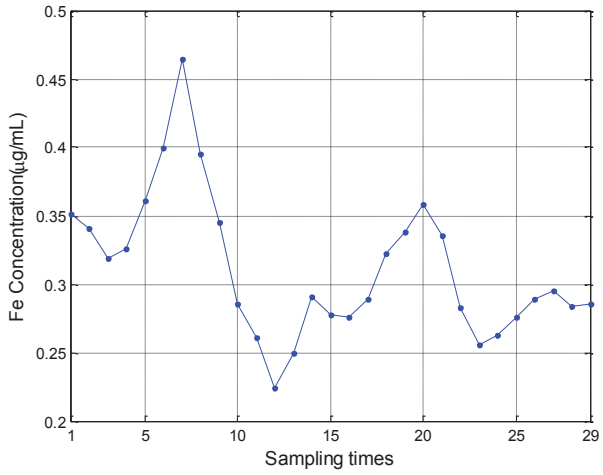

Fig. 4 *Fe concentration time series.*

In order to generate the training samples, we choose $(Fe_i, Fe_{i+1}, \cdots, Fe_{i+5})$ to be an input vector IV_i, $i = 1, \cdots, 135$, and Fe_{i+6} to be the corresponding output. Thus, we can get 135 couples of samples which can be denoted as $\{IV_i, Fe_{i+6}\}_{i=1}^{135}$. Then, the former 80 couples are selected to train the CSDPNN and the left 55 couples are used to test the CSDPNN. The topological structure of the CSDPNN is $1 - 30 - 1$, the error goal is set to 0.1, and the max iteration number is set to 1000. To make comparisons, PNN and ANN are also utilized to predict the same time series. We choose $(Fe_i, Fe_{i+1}, \cdots, Fe_{i+5})$to be an input function IF_i, $i = 1, \cdots, 135$ and Fe_{i+6} to be the corresponding output. Then, we can get 135 couples of samples which can be denoted as $\{IF_i, Fe_{i+6}\}_{i=1}^{135}$. As the same with CSDPNN, the former 80 couples are used to train the PNN model, and the left 55 couples are used to test the PNN. The parameters for PNN are set as the same with CSDPNN. And the topological structure of the ANN is again $6 - 30 - 1$. The prediction results are presented in Fig. 5.

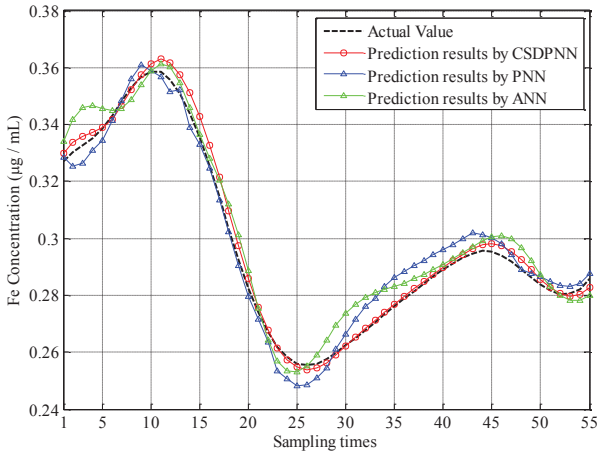

Fig. 5 *Fe concentration time series prediction results, where CSDPNN is with 1 input node.*

Since CSDPNN can have multi-inputs, we also utilized a multi-inputs CSDPNN with structure of 2-30-1 to predict the same Fe concentration time series. In this situation, the inputs of CSDPNN is a matrix with 2 rows, with each row as an input. Given an input vector as in the single input mode, such as $\mathbf{x} = [x_1, x_2, x_3, \cdots, x_5, x_n]$, with an embedded dimension m, $\mathbf{x}$ can be converted into a matrix

$$\mathbf{M} = \begin{bmatrix} x_1, x_2, \cdots, x_m \\ x_2, x_3, \cdots, x_{m+1} \\ \vdots \quad \vdots \quad \vdots \\ x_{n-m+1}, x_3, \cdots, x_n \end{bmatrix} \tag{18}$$

The matrix $\mathbf{M}$ is used as an input matrix of CSDPNN. In this paradigm, we set $m = 5$ to construct a 2-row input matrix from a vector with length of 6 used as input in the single input mode. Then, the training samples for multi-inputs

CSDPNN can be denoted as $\{\mathbf{M}_i, Fe_{i+6}\}_{i=1}^{135}$, where $\mathbf{M}_i$ is the *i-th* input matrix. The prediction results are shown in Fig. 6, and the prediction errors are presented in Tab. II.

	Absolute relative error	
	Max	Average
CSDPNN (1 input node)	2.27%	0.74%
CSDPNN (2 input nodes)	**1.99%**	**0.53%**
PNN	3.62%	1.52%
ANN	4.45%	1.77%

Tab. II *Fe concentration time series prediction results, where CSDPNN is with 2 input node.*

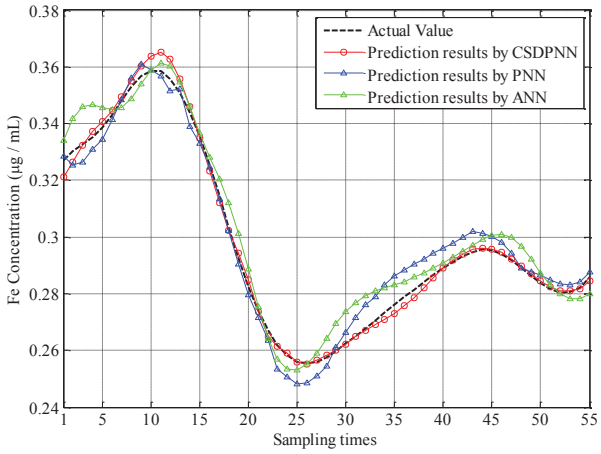

Fig. 6 *Fe concentration time series prediction results, where CSDPNN is with 2 input nodes.*

The results presented in Fig. 5, Fig. 6 and Tab. II indicate that CSDPNN has better performance than PNN and ANN in this paradigm. Moreover, since multi-inputs CSDPNN has a slightly smaller max relative absolute error and a mean relative absolute error than single-input CSDPNN, we conclude that the multi-inputs CSDPNN performs better. However, it also should be mentioned that the training of CSDPNN with multi-inputs costs more time.

6. Discussion and Conclusion

This paper proposed a convolution sum discrete process neural network (CSDPNN) for time series prediction. CSDPNN utilizes the discrete samples as inputs directly, and adopts the convolution sum operation to deal with the time accumulation effects in the time series. Compared with the PNN with continuous functions as

inputs, CSDPNN need not to fit the discrete samples to get the input functions, either to expand the input functions based on orthogonal basis functions, which can reduce precision loss and get higher precision. The theoretical analysis given in the paper shows that CSDPNN has approximation capacity between ANN and PNN. And the results of chaos time series prediction and iron concentration time series prediction proves that CSDPNN can get higher precision than PNN and ANN given the same conditions, which meets the theoretical expects. Also, CSDPNN with multi-inputs performs better than CSDPNN with single input.

Acknowledgment

This research was supported by the National Natural Science Foundation of China under Grant No.60939003, and the Harbin City Key Technologies R&D Program under Grant No. 2011AA1BG059.

References

[1] COSKUN O., ZUTURK C., SUNAR F., KARABOGA D. The Artificial Bee Colony Algorithm in Training Artificial Neural Network for Oil Spill Detection. *Neural Network World.* 2011, 21(6), pp. 473−492.

[2] DING G., ZHONG S.S. Thermal Equilibrium Temperature Prediction Based on Process Neural Network. *Journal of Astronautics (China).* 2006, 27(3), pp. 645−650.

[3] DING G., ZHONG S.S. Time Series Prediction by Parallel Feedforward Process Neural Network with Time-varied Input and Output Functions. *Neural Network World.* 2005, (15)2, pp. 137−147.

[4] HAGAN M.T., MENHAJ M. Training Feedforward Networks With the Marquardt Algorithm. *IEEE Transactions on Neural Networks.* 1994, 5(6), pp. 989−993, doi: 10.1109/72.329697.

[5] HE X.G., LIANG J.Z., XU S.H. Learning Algorithms of Process Neural Networks Based on Orthogonal Function Basis Expansion. *Chinese Journal of Computers.* 2004, (27)5, pp. 645−650.

[6] HE X.G., LIANG J.Z. Some Theoretical Issues on Process Neural Networks. *Engineering Science.* 2000, 12(2), pp. 40−44.

[7] REZA PEYGHAMI M., KHANDUZI R. Novel MLP Neural Network With Hybrid Tabu Search Algorithm. *Neural Network World.* 2013, 23(2), pp. 255−270.

[8] ZHONG S.S., CUI Z.Q., WANG T.C. Prediction of Engine Gas Path Parameter Deviation Based on Fractional Aggregation Process Neural Network. *Computer Integrated Manufacturing Systems.* 2013, 19(5), pp. 1071−1077.

[9] ZHONG S.S., DING G., SU D.Z. Parallel Feedforward Process Neural Network with Time-varying Input and Output Functions. *Lecture Notes in Computer Science.* 2005, 3496(1), pp. 473−478.

[10] ZHONG S.S., LI Y., DING G., LIN L. Continuous Wavelet Process Neural Network and Its Application. *Neural Network World.* 2007, 17(5), pp. 483−495.