International Journal of Control and Automation
Vol.7, No.8 (2014), pp. 117-130
http://dx.doi.org/10.14257/ijca.2014.7.8.11

A Sliding-Window Modeling Approach for Neural Network

Xiao Laisheng

Guangdong Ocean University, Zhanjiang, 524088, China
xiaolaisheng@163.com

Abstract

A sliding-window modeling approach of neural network (SWMANN) was presented. The basic idea is that training data for neural network should be reconstructed by means of a slide-window way to build input and target samples. It means that the output is determined not only by the current input, but also by the past input, which could better follow the dynamic change and development of real systems. In this paper, SWMANN was introduced. A detailed theoretical derivation was described for its implementation by taking wavelet neural network as a modeling tool. Moreover, SWMANN was compared with classical modeling approach of neural network (CMANN). Theoretical results indicated that CMANN is a special case of SWMANN when sliding-window parameters are selected as 1. Therefore, compared with CMANN, SWMANN presented in this paper is a more general form.

Keywords: *neural network; sliding-window modeling; wavelet; dynamical system*

1. Introduction

A dynamical system is one that evolves over time through a set of explicit rules, and has a typical structure of multiple inputs and multiple outputs. Generally, it is composed of components interacting through a set of explicit rules. The interactions encapsulated in these rules dictate how the states of the components evolve in time, and so the notion of time evolution is critical when thinking about such a system [1]. Current outputs of dynamical systems are not only determined by its current inputs but also related to its previous or past inputs.

Modeling a dynamical system is of crucial importance for investigating its behavior. Up to now, experts and scholars in this field have done a massive exploration and invented a series of modeling approaches. The common used modeling approaches for dynamical systems are summed up as: 1. Mathematical Modeling [1-3]; 2. Dynamic linearization technique (DLT) [4]; 3. Grey modeling approach [5]; 4. Multi-swarm optimizer [6]; 5. Bayesian network and ant colony algorithm [7]; 6. Wavelet spectrum analysis [8]; 7. Neural network approaches, including: BP neural network [9], Recurrent fuzzy neural network with support vector regression [10], Recurrent interval-valued fuzzy neural network [11], Dynamic fuzzy neural networks [12, 13], Fuzzy wavelet neural network modeling [14], *etc*. At present, neural network approaches are most widely used for dynamical system modeling.

With regard to neural network modeling for dynamic systems, different scholars studied different objects. For examples, Adnan hashman [15] described a credit risk evaluation system that uses supervised neural network models based on the back propagation learning algorithm. In Ercan Ozgan' study [16], the Marshall Stability of asphalt concrete under varying temperature and exposure times was modeled by using artificial neural network. Anqi Cui, *etc.*, [17], presented an Elman neural network-based model to predict detergents' anti-germ performance and ingredient levels, respectively. Erkam Guresen, *etc.*, [18] evaluated the

ISSN: 2005-4297 IJCA

effectiveness of neural network models which are known to be dynamic and effective in stock-market predictions. Paper [19] presented a methodology based on neural networks in order to replace the use of net radiometers (expensive tools) by modeling the relationships between the net radiation and meteorological variables measured in meteorological stations. An artificial neural network [20] based on multilayer perceptrons with back propagation algorithm was used to approximate and interpret the complex input/output relationship, essentially to understand the breakthrough times in engineered floodplain filtration (EFF) system. But modeling approaches they used were basically the same, which can be summarized as follows.

1. Choose a specific dynamic system as a modeling object;
2. Analyze characteristics of the dynamic system, select some parameters as its independent input variables and some parameters as its output variables for the system;
3. Construct neural network structure, where the number of input terminals of the neural network should be equal to that of the independent input variables of the dynamic system, and the number of output terminals of the neural network should be equal to that of the output variables of the dynamic system;
4. Train neural network with input/output samples of the dynamic system in turn, until it reaches precision requirement;
5. When training is finished, the neural network becomes to be the model of the dynamic system which is expected.

In order to distinguish it from others, the modeling approach described above could be named as classical modeling approach of neural network (CMANN).

In CMANN, there is an obvious drawback that current outputs of neural network are only determined by its current inputs. However, Current outputs of dynamical systems are not only determined by its current inputs but also related to its previous or past inputs, which is just ignored by CMANN. Aiming at this case, this paper proposed a sliding-window modeling approach for neural network (SWMANN). The basic idea is that training data of neural network should be reconstructed by means of a slide-window way to build input and target samples. It means that the output is determined not only by the current input, but also by the past input, which could better follow the dynamic change and development of real systems.

The rest of this paper is organized as follows. Section 2 introduces the basic principle of SWMANN in detailed. In Section 3, detailed theoretical derivations of CMANN and SWMANN are described by taking wavelet neural network as a modeling tool in order to discuss the implementation of SWMANN. In Section 4, several principles for constructing dynamical system models with SWMANN are put forward. In Section 5, relationship between SWMANN and CMANN is discussed. Finally, conclusions are given in Section 6.

2. Description of SWMANN

A sliding-window modeling approach for neural network (SWMANN) is a new approach to model and predict dynamical systems. Neural network structure of SWMANN is shown in Figure 1.

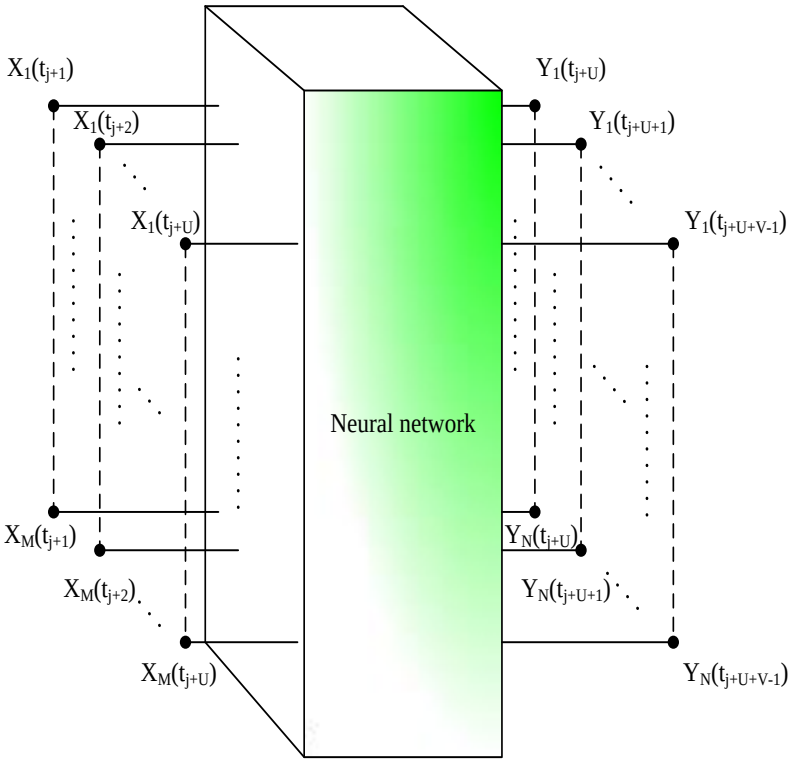

Figure 1. Neural Network Structure of SWMANN

In Figure 1, assume that input vectors and output vectors of samples of a dynamic system are $X(t_1), X(t_2), \cdots, X(t_k)$ and $Y(t_1), Y(t_2), \cdots, Y(t_k)$ respectively at time $t_1, t_2, \cdots, t_k$. Where, $X(t_1), X(t_2), \cdots, X(t_k)$ are all m-dimensional vectors, and $Y(t_1), Y(t_2), \cdots, Y(t_k)$ are all n-dimensional vectors. Now, two sliding-window modeling parameters are introduced, sliding-window input parameter marked as U and sliding-window output parameter marked as V, where, U and V should all be positive integers and should be greater than 1. The number of input terminals of neural network is $M \times U$ and the number of its output terminals is $N \times V$. The neural network is a $(M \times U + N \times V)$ dimensional model.

Modeling process of SWMANN is described as follows. Step one, take $X(t_1), X(t_2), \cdots, X(t_U)$ as its input vectors and $Y(t_U), Y(t_{U+1}), \cdots, Y(t_{U+V-1})$ as its target vectors to train the neural network. Step two, take $X(t_2), X(t_3), \cdots, X(t_{U+1})$ as its input vectors and $Y(t_{U+1}), Y(t_{U+2}), \cdots, Y(t_{U+V})$ as its target vectors to train it, and so on. Until the last step, take $X(t_{k-U-V+2}), X(t_{k-U-V+3}), \cdots, X(t_{k-V+1})$ as its input vectors and $Y(t_{k-V+1}), Y(t_{k-V+2}), \cdots, Y(t_k)$ as its target vectors to train it. After the first round of training is finished, if network precision does not reach requirement, then loop from step one, until network precision reaches requirement or maximum number of iterations is achieved. Its modeling process is shown in Figure 2.

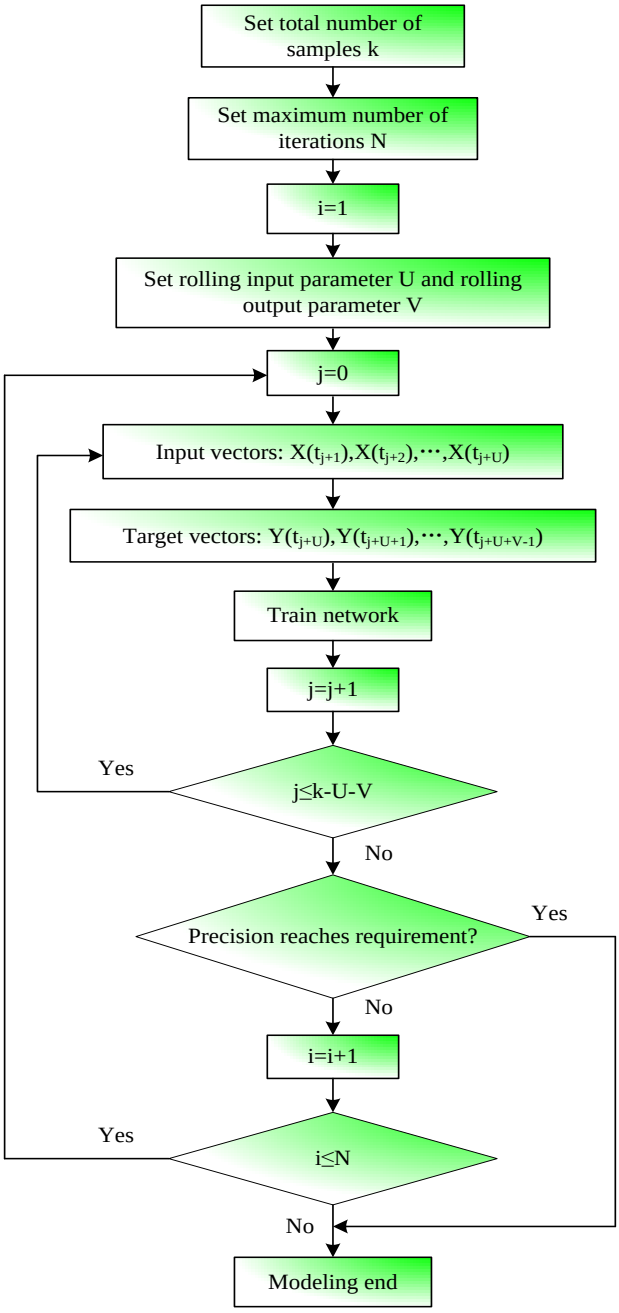

Figure 2. Modeling Process with SWMANN

Behaviors of a dynamic system can be predicted based on SWMANN. Its predicting method is described as follows. At first, a sliding-window model for the dynamic system of multiple inputs and multiple outputs should be set up. Then sliding window prediction can be done for the dynamic system based on this model. When sliding-window input parameter is set as U and sliding-window parameter is set as V, predicting process of SWMANN is shown in Figure 3.

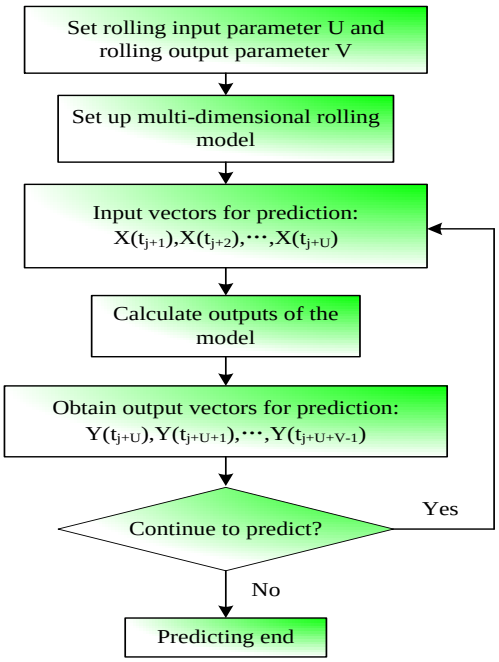

Figure 3. Predicting Process with SWMANN

Once sliding-window model for neural network has been set up, one may use the sliding-window predicting method to continuously predict its behavior. For any input vectors, corresponding predicting outputs of the model can be obtained through calculation.

3. Implementation of SWMANN

In the following description, implementation of SWMANN is studied in detailed by taking wavelet neural network as an example. Before implementation of SWMANN, a detailed theoretical derivation for CMANN was described in order to study the relationship between SWMANN and CMANN.

3.1. Wavelet Neural Network with CMANN

3.1.1. Basic Conception of Wavelet

Wavelet analysis is a new mathematics branch developed in recent 20 years. Its occurrence provides a powerful tool for non-stationary signal analysis and processing. Wavelet transform has different resolution in different positions of time and frequency plane and is a kind of multi-resolution signal analysis method. Its concept is retold briefly as follows.

Definition 1: Assume $\varphi(t) \in L^2(R)$, if

$$\int_{-\infty}^{+\infty} \varphi(t)dt = 0$$

Where, $\varphi(t)$ is known as a wavelet, also often referred to as mother wavelet or basic wavelet.

Definition 2: Move and stretch wavelet $\varphi(t)$, a family of functions can be obtained

$$\varphi_{u,s}(t) = \frac{1}{\sqrt{s}}\varphi(\frac{t-u}{s}) \qquad s > 0, u \in R$$

Where, $\varphi_{u,s}(t)$ is called as wavelet basis function of wavelet $\varphi(t)$, s is known as scale parameter, u is referred to as translation parameter.

Essence of wavelet transform is to express or approximate a function by means of a family of functions constructed through moving and stretching a basic wavelet function.

3.1.2. Structure of Wavelet Neural Network with CMANN

Wavelet neural network is a feed forward network model with wavelet basis functions as excitation function of neurons. Its basic strategy is that using the minimization principle of error function changes the shape and scale of wavelet basis in order to adjust the weights and thresholds of the network. Structure of wavelet neural network is a three-layer of feed forward network, as shown in Figure 4.

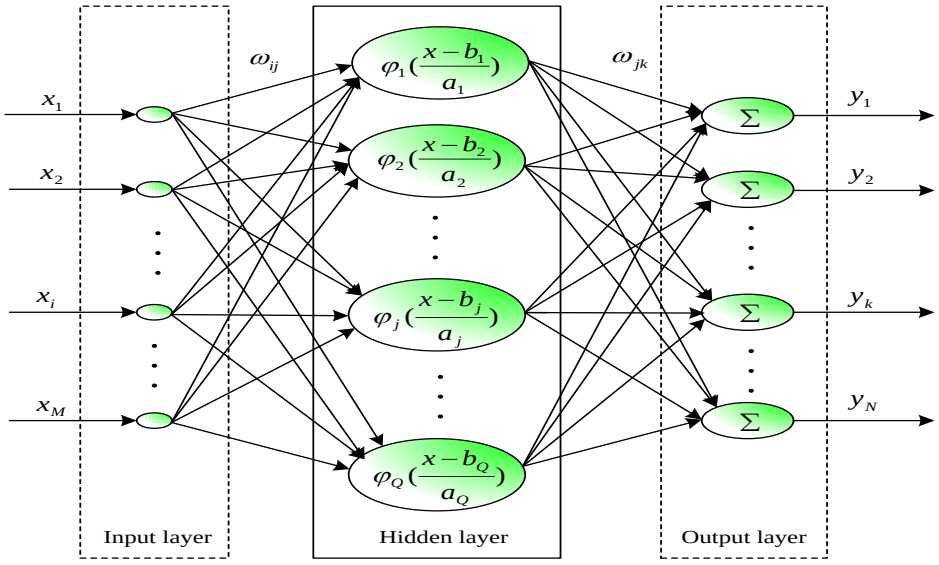

Figure 4. Structure of Wavelet Neural Network with CMANN

In Figure 4, assume that M is total number of nodes in input layer, $i = 1,2,\cdots,M$; Q is total number of nodes in hidden layer, $j = 1,2,\cdots,Q$; N is total number of nodes in output layer, $k = 1,2,\cdots,N$; P is total number of input samples, $p = 1,2,\cdots,P$; $\{x_i^p\}$ are inputs of network, $\{y_k^p\}$ are outputs of network, $\{d_k^p\}$ are their corresponding target outputs; ω_{ij} are connection weights between neurons in input layer and neurons in hidden layers, ω_{jk} are connection weights between neurons in hidden layer and neurons in output layers; a_j is scale parameter and b_j is translation parameter of wavelet. Then input of jth neuron of wavelet

basis can be expressed as $s_j^p = \sum_{i=1}^{M} \omega_{ij} x_i^p$, its output is $t_j^p = \varphi(\frac{s_j^p - b_j}{a_j})$, output of *kth* level component in output layer is $y_k^p = \sum_{j=1}^{Q} \omega_{jk} t_j^p$, total error function is defined as $E = \frac{1}{P}\sum_{p=1}^{P} E^p = \frac{1}{2P}\sum_{p=1}^{P}\sum_{k=1}^{N}(d_k^p - y_k^p)^2$.

3.1.3. Learning Algorithm of Wavelet Neural Network with CMANN

Learning algorithm of wavelet neural network is to use BP algorithm to adjust parameters of network ω_{ij}, ω_{jk}, a_j, b_j based on error function. Its adjustment process is divided into two steps: The first stage is a feed-forward process. Calculation is put forward from input layer to output layer of the network, including calculating outputs of each layer according to input sample data, and finally find out outputs of output layer in the network. The second stage is a back propagation process. Parameters of network are fixed from output layer of the network back to its input layer. These two processes repeats alternately until the total error E meet the requirement of the network. Concrete steps of learning algorithm are described as follows.

Step one, Initialize network weights, thresholds, scale and translation parameters of wavelet function.

Give corresponding initial values for each of the following parameters: scale parameter a_j and translation parameter b_j of the wavelet, weights ω_{ij} and ω_{jk} of the network, learning rate η and momentum factor λ.

Set sample counter $p = 1$ and number of iterations $n = 1$, and Assume that maximum number of iterations is N.

Step two, Input learning samples $\{x_i^p\}$ and corresponding desired outputs $\{d_k^p\}$, calculate outputs of hidden layer $\{t_j^p\}$ and outputs of output layer $\{y_k^p\}$.

Inputs of hidden layer are

$$s_j^p = \sum_{i=1}^{M} \omega_{ij} x_i^p, \quad j = 1,2,\cdots,Q \tag{1-a}$$

Outputs of hidden layer are

$$t_j^p = \varphi(\frac{s_j^p - b_j}{a_j}) = \varphi(\frac{\sum_{i=1}^{M} \omega_{ij} x_i^p - b_j}{a_j}), \quad j = 1,2,\cdots,Q \tag{2-a}$$

Outputs of output layer are

$$y_k^p = \sum_{j=1}^{Q} \omega_{jk} t_j^p, \quad k = 1,2,\cdots,N \tag{3-a}$$

In equations (1-a),(2-a), and (3-a), x_i^p is input of input layer, s_j^p is input of hidden layer, t_j^p is output of hidden layer, y_k^p is output of output layer, $\varphi(\cdot)$ is wavelet basis function.

Step three, Calculate target error and gradient vectors.

Define target error function E^p as

$$E^p = \frac{1}{2}\sum_{k=1}^{N}(d_k^p - y_k^p)^2 \tag{4-a}$$

Energy function gradient respectively are

$$\delta_{ij}^p = \frac{\partial E^p}{\partial \omega_{ij}} = \frac{\partial E^p}{\partial y_k^p}\frac{\partial y_k^p}{\partial t_j^p}\frac{\partial t_j^p}{\partial \omega_{ij}}$$

$$= -\sum_{k=1}^{N}(d_k^p - y_k^p)\omega_{jk}\varphi'\left(\frac{s_j^p - b_j}{a_j}\right)\frac{x_i^p}{a_j} \tag{5-a}$$

Where, ω_{ij} can affect all level component of output layer, so

$$\frac{\partial E^p}{\partial y_k^p} = -\sum_{k=1}^{N}(d_k^p - y_k^p)$$

$$\delta_{jk}^p = \frac{\partial E^p}{\partial \omega_{jk}} = \frac{\partial E^p}{\partial y_k^p}\frac{\partial y_k^p}{\partial \omega_{jk}} = -(d_k^p - y_k^p)t_j^p = -(d_k^p - y_k^p)\varphi\left(\frac{s_j^p - b_j}{a_j}\right) \tag{6-a}$$

But, ω_{ij} can only affect kth level component of output layer, so

$$\frac{\partial E^p}{\partial y_k^p} = -(d_k^p - y_k^p)$$

$$\delta_{aj}^p = \frac{\partial E^p}{\partial a_j} = \frac{\partial E^p}{\partial y_k^p}\frac{\partial y_k^p}{\partial t_j^p}\frac{\partial t_j^p}{\partial a_j}$$

$$= -\sum_{k=1}^{N}(d_k^p - y_k^p)\omega_{jk}\varphi'\left(\frac{s_j^p - b_j}{a_j}\right)\left(\frac{s_j^p - b_j}{a_j^2}\right) \tag{7-a}$$

$$\delta_{bj}^p = \frac{\partial E^p}{\partial b_j} = \frac{\partial E^p}{\partial y_k^p}\frac{\partial y_k^p}{\partial t_j^p}\frac{\partial t_j^p}{\partial b_j} = -\sum_{k=1}^{N}(d_k^p - y_k^p)\omega_{jk}\varphi'\left(\frac{s_j^p - b_j}{a_j}\right)\left(-\frac{1}{a_j}\right) \tag{8-a}$$

Step four, Error back propagation and modify network parameters.

Computational formulas are

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \eta\delta_{ij}^p + \lambda[\omega_{ij}(t) - \omega_{ij}(t-1)] \tag{9-a}$$

$$\omega_{jk}(t+1) = \omega_{jk}(t) - \eta\delta_{jk}^p + \lambda[\omega_{jk}(t) - \omega_{jk}(t-1)] \tag{10-a}$$

$$a_j(t+1) = a_j(t) - \eta\delta_{aj}^p + \lambda[a_j(t) - a_j(t-1)] \tag{11-a}$$

$$b_j(t+1) = b_j(t) - \eta\delta_{bj}^p + \lambda[b_j(t) - b_j(t-1)] \tag{12-a}$$

Step five, Input next sample, namely set $p = p + 1$. If $p \le P$, then go to **Step two**.

Step six, Calculate total error of the network

$$E = \frac{1}{P}\sum_{p=1}^{P} E^{p} = \frac{1}{2P}\sum_{p=1}^{P}\sum_{k=1}^{N}(d_k^p - y_k^p)^2 \tag{13-a}$$

And judge if E is smaller than preset progress value $\varepsilon(\varepsilon > 0)$. If $E < \varepsilon$ or $n > N$, stop the learning of network, otherwise, set $n = n + 1$ and $p = 1$, go to **Step two**.

In general, Morlet wavelet is used as wavelet basis function. Its expressions are

$$\varphi(t) = \cos(1.75t)\exp(-\frac{t^2}{2})$$

$$\frac{\partial\varphi(t)}{\partial t} = -[x\cos(1.75x) + 1.75\sin(1.75x)]e^{-\frac{x^2}{2}}$$

3.2. Wavelet Neural Network with SWMANN

When SWMANN is employed, the network structure and learning algorithm of wavelet neural network are needed to be improved.

3.2.1. Structure of Wavelet Neural Network with SWMANN

When SWMANN is used, the structure of wavelet neural network is shown in Figure 5.

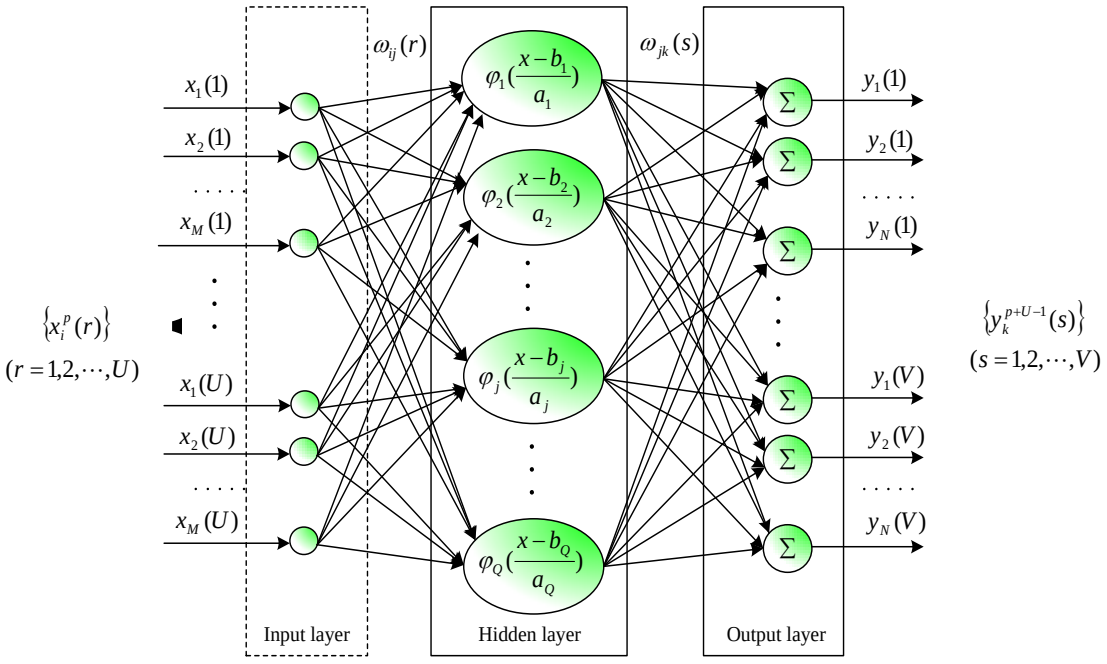

Figure 5. Structure of Wavelet Neural Network with SWMANN

In Figure 5, U is sliding-window input parameter, V is sliding-window output parameter. In this case, input weights of network ω_{ij} are extended equally to the number of $\omega_{ij} \times U$, which are marked as $\omega_{ij}(r), r = 1,2,\cdots,U$, and output weights of network ω_{jk} are extended

equally to the number of $\omega_{jk} \times V$, which are marked as $\omega_{jk}(s), s = 1,2,\cdots V$. The meanings of other parameters are as the same as in Figure 4.

3.2.2. Learning Algorithm of Wavelet Neural Network with SWMANN

When SWMANN is used, the learning algorithm of wavelet neural network is described as follows.

Step one, Initialize network weights, thresholds, scale and translation parameters of wavelet function.

Give corresponding initial values for each of the following parameters: scale parameter a_j and translation parameter b_j of the wavelet, weights of the network $\omega_{ij}(r), r = 1,2,\cdots,U$ and $\omega_{jk}(s), s = 1,2,\cdots V$, learning rate η and momentum factor λ.

Set sample counter $p = 1$ and number of iterations $n = 1$, and provided maximum number of iterations is N.

Step two, Input learning samples $\{x_i^p(r)\}, r = 1,2,\cdots,U$ and corresponding sliding-window desired outputs $\{d_k^{p+U-1}(s)\}, s = 1,2,\cdots,V$, and calculate outputs of hidden layer $\{t_j^p\}$ and sliding-window outputs of output layer $\{y_k^{p+U-1}(s)\}, s = 1,2,\cdots,V$.

Inputs of hidden layer are

$$s_j^p = \sum_{i=1}^{M}(\sum_{r=1}^{U}\omega_{ij}(r)x_i^p(r)), \quad j = 1,2,\cdots,Q \tag{1-b}$$

Outputs of hidden layer are

$$t_j^p = \varphi(\frac{s_j^p - b_j}{a_j}) = \varphi(\frac{\sum_{i=1}^{M}(\sum_{r=1}^{U}\omega_{ij}(r)x_i^p(r)) - b_j}{a_j}), \quad j = 1,2,\cdots,Q \tag{2-b}$$

Sliding-window outputs of output layer are

$$y_k^{p+U-1}(s) = \sum_{j=1}^{Q}\omega_{jk}(s)t_j^p, \quad k = 1,2,\cdots,N \ . \ s = 1,2,\cdots V \tag{3-b}$$

In equations (1-b),(2-b), and (3-b), $x_i^p(r)$ is input of input layer, s_j^p is input of hidden layer, t_j^p is output of hidden layer, $y_k^{p+U-1}(s)$ is sliding-window output of output layer, $\varphi(\cdot)$ is wavelet basis function.

Step three, Calculate target error and gradient vectors.

Define target error function E^p as

$$E^p = \frac{1}{2}\sum_{k=1}^{N}\sum_{s=1}^{V}(d_k^{p+U-1}(s) - y_k^{p+U-1}(s))^2 \tag{4-b}$$

Where, $\{d_k^{p+U-1}(s)\}$ is sliding-window desired outputs of output layer, $y_k^{p+U-1}(s)$ is sliding-window calculated outputs of output layer.

Energy function gradient respectively are

$$\delta_{ij}^{p}(r) = \frac{\partial E^{p}}{\partial \omega_{ij}(r)} = \frac{\partial E^{p}}{\partial y_{k}^{p+U-1}(s)} \frac{\partial y_{k}^{p+U-1}(s)}{\partial t_{j}^{p}} \frac{\partial t_{j}^{p}}{\partial \omega_{ij}(r)}$$

$$= -\sum_{k=1}^{N}\sum_{s=1}^{V}(d_{k}^{p+U-1}(s) - y_{k}^{p+U-1}(s))\omega_{jk}(s)\varphi'(\frac{s_{j}^{p} - b_{j}}{a_{j}})\frac{x_{i}^{p}(r)}{a_{j}} \quad (5\text{-b})$$

Where, $\omega_{ij}(r)$ can affect all level component of output layer, so

$$\frac{\partial E^{p}}{\partial y_{k}^{p+U-1}} = -\sum_{k=1}^{N}\sum_{s=1}^{V}(d_{k}^{p+U-1}(s) - y_{k}^{p+U-1}(s))$$

$$\delta_{jk}^{p}(s) = \frac{\partial E^{p}}{\partial \omega_{jk}(s)} = \frac{\partial E^{p}}{\partial y_{k}^{p+U-1}(s)} \frac{\partial y_{k}^{p+U-1}(s)}{\partial \omega_{jk}(s)} = -(d_{k}^{p+U-1}(s) - y_{k}^{p+U-1}(s))t_{j}^{p}$$

$$= -(d_{k}^{p+U-1}(s) - y_{k}^{p+U-1}(s))\varphi(\frac{s_{j}^{p} - b_{j}}{a_{j}}) \quad (6\text{-b})$$

But, $\omega_{ij}(s)$ can only affect kth level component of output layer, so

$$\frac{\partial E^{p}}{\partial y_{k}^{p+U-1}} = -(d_{k}^{p+U-1}(s) - y_{k}^{p+U-1}(s))$$

$$\delta_{aj}^{p} = \frac{\partial E^{p}}{\partial a_{j}} = \frac{\partial E^{p}}{\partial y_{k}^{p+U-1}(s)} \frac{\partial y_{k}^{p+U-1}(s)}{\partial t_{j}^{p}} \frac{\partial t_{j}^{p}}{\partial a_{j}}$$

$$= -\sum_{k=1}^{N}\sum_{s=1}^{V}(d_{k}^{p+U-1}(s) - y_{k}^{p+U-1}(s))\omega_{jk}\varphi'(\frac{s_{j}^{p} - b_{j}}{a_{j}})(\frac{s_{j}^{p} - b_{j}}{a_{j}^{2}}) \quad (7\text{-b})$$

$$\delta_{bj}^{p} = \frac{\partial E^{p}}{\partial b_{j}} = \frac{\partial E^{p}}{\partial y_{k}^{p+U-1}(s)} \frac{\partial y_{k}^{p+U-1}(s)}{\partial t_{j}^{p}} \frac{\partial t_{j}^{p}}{\partial b_{j}}$$

$$= -\sum_{k=1}^{N}\sum_{s=1}^{V}(d_{k}^{p+U-1}(s) - y_{k}^{p+U-1}(s))\omega_{jk}\varphi'(\frac{s_{j}^{p} - b_{j}}{a_{j}})(-\frac{1}{a_{j}}) \quad (8\text{-b})$$

Step four, Error back propagation and modify network parameters.

Computational formulas are

$$\omega_{ij}(r)(t+1) = \omega_{ij}(r)(t) - \eta\delta_{ij}^{p}(r) + \lambda[\omega_{ij}(r)(t) - \omega_{ij}(r)(t-1)] \quad (9\text{-b})$$

$$\omega_{jk}(s)(t+1) = \omega_{jk}(s)(t) - \eta\delta_{jk}^{p}(s) + \lambda[\omega_{jk}(s)(t) - \omega_{jk}(s)(t-1)] \quad (10\text{-b})$$

$$a_{j}(t+1) = a_{j}(t) - \eta\delta_{aj}^{p} + \lambda[a_{j}(t) - a_{j}(t-1)] \quad (11\text{-b})$$

$$b_{j}(t+1) = b_{j}(t) - \eta\delta_{bj}^{p} + \lambda[b_{j}(t) - b_{j}(t-1)] \quad (12\text{-b})$$

Step five, Input next sample, namely set $p = p + 1$. If $p \le P$, then go to **Step two**.

Step six, Calculate total error of the network

$$E = \frac{1}{P}\sum_{p=1}^{P}E^{p} = \frac{1}{2P}\sum_{p=1}^{P}\sum_{k=1}^{N}\sum_{s=1}^{V}(d_{k}^{p+U-1}(s) - y_{k}^{p+U-1}(s))^{2} \quad (13\text{-b})$$

And judge if E is smaller than preset progress value $\varepsilon(\varepsilon > 0)$. If $E < \varepsilon$ or $n > N$, stop the learning of network, otherwise, set $n = n + 1$ and $p = 1$, go to **Step two**.

4. Modeling and Predicting Dynamical Systems with SWMANN

Assume that the number of input terminals of a dynamical system is M and its number of output terminals is N. At time t, its input vectors are $X(t) = [X_1(t), X_2(t) \cdots, X_M(t)]$, and its corresponding output vectors are $Y(t) = [Y_1(t), Y_2(t) \cdots, Y_N(t)]$. Where, $X_1(t), X_2(t) \cdots, X_M(t)$ are M independent variables of input terminals, $Y_1(t), Y_2(t) \cdots, Y_N(t)$ are N corresponding independent outputs of output terminals, t represents time sequence. Structure of dynamical system is shown in Figure 6.

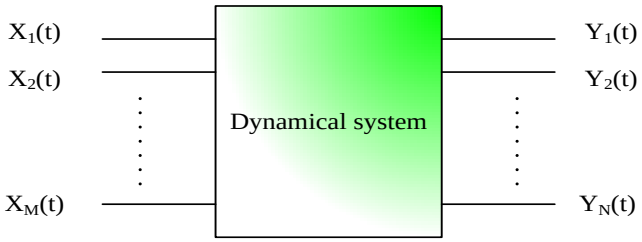

Figure 6. Structure of Dynamical System

For the dynamic system shown in Figure 6, if SWMANN is used, the following several principles should be followed.

1. For properties of data, input/output samples of dynamic system should be data in time sequence;

2. For structure of network, assume that sliding-window input parameter is set as U and sliding-window output parameter is set as V, where, U and V should all be positive integers which should be greater than 1. Then the number of input terminals of neural network is $M \times U$ and the number of its output terminals is $N \times V$. The neural network is a $(M \times U + N \times V)$ dimensional model;

3. For data processing, it is a procedure of continuous sliding-window forward in time;

4. For training of network, sliding-window input vectors should be m-dimensional vectors each time and their number should be U. And sliding window desired output vectors should be n-dimensional vectors and their number should be V;

5. For prediction of network, similarly, sliding window input vectors should be m-dimensional vectors each time and their number should be U. And sliding-window output vectors obtained by calculation should be n-dimensional vectors and their number should be V;

6. Especially, in the modeling process mentioned above, if set $U = 1$ and $V = 1$, then SWMANN becomes into CMANN (see Discussion in Section 5 of this paper).

In a word, compared with CMANN, SWMANN is sliding-window modeling and predicting approach, in which the output is determined not only by the current input, but also by the past input. Therefore, it could better follow the dynamic change and development of real systems.

5. Discussions

From the detailed theoretical derivation, the implementation of SWMANN in Section 3 of this paper, it is easily known that when $U = 1$ and $V = 1$, the structure of wavelet neural network in Figure 5 becomes into one in Figure 4, and all equations (1-b), (2-b), …, (13-b) in Section 3.2.2 become into equations (1-a), (2-a), …, (13-a) in Section 3.1.3. In other words, SWMANN becomes into CMANN, so CMANN is a special case of SWMANN when sliding-window modeling parameters are selected as 1. Therefore, compared with CMANN, SWMANN presented in this paper is a more general form.

6. Conclusions

To address simulating and modeling of dynamical systems with multiple inputs and outputs, a sliding-window modeling approach for neural network (SWMANN) was presented, in which neural network was employed to model and predict dynamical systems. In SWMANN, training data of neural network should be reconstructed by means of a sliding-window way to build input and target samples. It means that the output is determined not only by the current input, but also by the past input, which could better follow the dynamic change and development of real systems. Theoretical results indicated that CMANN is a special case of SWMANN when sliding-window modeling parameters are selected as 1. Therefore, compared with CMANN, SWMANN presented in this paper is a more general form.

References

[1] S. Daun, J. Rubin, Y. Vodovotz and G. Clermont, "Equation-based models of dynamic biological systems", Journal of Critical Care, vol. 23, **(2008)**, pp. 585–594.
[2] N. TAKAHASHI, H. TAKENO and Y. OHSAWA, "Identification of Power System Dynamics Due to Combined Use of Mathematical Model and Neural Network", Electrical Engineering in Japan, vol. 138, no. 1, **(2002)**, 42-48.
[3] J.-S. Pei, J. P. Wright and A. W. Smyth, "Mapping polynomial fitting into feedforward neural networks for modeling nonlinear dynamic systems and beyond", Comput. Methods Appl. Mech. Engrg., vol. 194, **(2005)**, pp. 4481-4505.
[4] Z. Hou and S. Jin, "Data-Driven Model-Free Adaptive Control for a Class of MIMO Nonlinear Discrete-Time Systems", IEEE TRANSACTIONS ON NEURAL NETWORKS, vol. 22, no. 12, **(2011)** December, pp. 2173-2187.
[5] X. Tian, M. Ju, C. Shao and Z. Fang, "Developing a new grey dynamic modeling system for evaluation of biology and pollution indicators of the marine environment in coastal areas", Ocean & Coastal Management, vol. 54, **(2011)**, pp. 750-759.
[6] B. Niu, Y. Zhu, X. He and H. Shen, "A multi-swarm optimizer based fuzzy modeling approach for dynamic systems processing", Neurocomputing, vol. 71, **(2008)**, pp. 1436-1448.
[7] J. Hua, L. Zhang, L. Ma and W. Liang, "An integrated safety prognosis model for complex system based on dynamic Bayesian network and ant colony algorithm", Expert Systems with Applications, vol. 38, **(2011)**, pp. 1431–1446.
[8] X. Jiang and S. Mahadevan, "Wavelet spectrum analysis approach to model validation of dynamic systems", Mechanical Systems and Signal Processing, vol. 25, **(2011)**, pp. 575–590.
[9] Y. Ma, M. Huang, J. Wan, Y. Wang, X. Sun and H. Zhang, "Prediction model of DnBP degradation based on BP neural network in AAO system", Bioresource Technology, vol. 102, **(2011)**, pp. 4410–4415.
[10] C.-F. Juang and C.-D. Hsieh, "A Locally Recurrent Fuzzy Neural Network With Support Vector Regression for Dynamic-System Modeling", IEEE Transactions on Fuzzy Systems, vol. 18, no. 2, **(2010)** April, pp. 261-273.
[11] C.-F. Juang, Y.-Y. Lin and R.-B. Huang, "Dynamic system modeling using a recurrent interval-valued fuzzy neural network and its hardware implementation", Fuzzy Sets and Systems, vol. 179, **(2011)**, pp. 83–99.
[12] X. Deng and X. Wang, "Incremental learning of dynamic fuzzy neural networks for accurate system modeling", Fuzzy Sets and Systems, vol. 160, **(2009)**, pp. 972–987.

[13] D. Lin, X. Wang, F. Nian and Y. Zhang, "Dynamic fuzzy neural networks modeling and adaptive backstepping tracking control of uncertain chaotic systems", Neurocomputing, vol. 73, **(2010)**, pp. 2873–2881.
[14] S. Yilmaz and Y. Oysal, "Fuzzy Wavelet Neural Network Models for Prediction and Identification of Dynamical Systems", IEEE TRANSACTIONS ON NEURAL NETWORKS, vol. 21, no. 10, **(2010)** October, pp.1599-1609.
[15] A. Khashman, "Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes", Expert Systems with Applications, vol. 37, **(2010)**, pp. 6233–6239.
[16] E. Ozgan, "Artificial neural network based modelling of the Marshall Stability ofasphalt concrete", Expert Systems with Applications, vol. 38, **(2011)**, pp. 6025–6030.
[17] A. Cui, H. Xu and P. Jia, "An Elman neural network-based model for predicting anti-germ performances and ingredient levels with limited experimental data", Expert Systems with Applications, vol. 38, **(2011)**, pp. 8186–8192.
[18] E. Guresen, G. Kayakutlu and T. U. Daim, "Using artificial neural network models in stock market index prediction", Expert Systems with Applications, vol. 38, **(2011)**, pp. 10389-10397.
[19] A. Geraldo-Ferreira, E. Soria-Olivas, J. Gómez-Sanchis, A. José Serrano-López, A. Velázquez-Blazquez and E. López-Baeza, "Modelling net radiation at surface using ''in situ'' netpyrradiometer measurements with artificial neural networks", Expert Systems with Applications, vol. 38, **(2011)**, pp. 14190–14195.
[20] S. K. Behera, E. R. Rene and H.-S. Park, "Neural network modeling of sorption of pharmaceuticals in engineered floodplain filtration system", Expert Systems with Applications, vol. 39, **(2012)**, pp. 6052–6060.

Author

Xiao Laisheng was born in Hubei, China, in 1962. He received the Master's Degree in computer application technology from Wuhan University, Wuhan, China, in 2003. He is currently an associate professor with Guangdong Ocean University, China.

He is the author and coauthor of over 30 scientific papers. His current research interests include the areas of computer networks, computational intelligence techniques and their applications.

