Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 429402, 7 pages
http://dx.doi.org/10.1155/2013/429402

Research Article

Memristive Chebyshev Neural Network and Its Applications in Function Approximation

Lidan Wang, Meitao Duan, and Shukai Duan

School of Electronic and Information Engineering, Southwest University, Chongqing 400715, China

Correspondence should be addressed to Shukai Duan; duansk@swu.edu.cn

Received 1 February 2013; Accepted 22 April 2013

Academic Editor: Chuandong Li

A novel Chebyshev neural network combined with memristors is proposed to perform the function approximation. The relationship between memristive conductance and weight update is derived, and the model of a single-input memristive Chebyshev neural network is established. Corresponding BP algorithm and deriving algorithm are introduced to the memristive Chebyshev neural networks. Their advantages include less model complexity, easy convergence of the algorithm, and easy circuit implementation. Through the MATLAB simulation results, we verify the feasibility and effectiveness of the memristive Chebyshev neural networks.

1. Introduction

Many researchers have further studied the approximation capacity of neural networks. Research indicates that there are some drawbacks in Back Propagation (BP) neural networks, such as slow convergence, getting easily into a local minimum, and difficulties in determining the numbers of hidden neurons. The Radical Basis Function (RBF) neural networks are superior to BP neural networks in approximation capacity because their learning rate is fast, but the lack of theoretical guidance in determining the central position of basis function makes us set the central position based on experience. We cannot ensure whether the performance of networks is the best or not. Especially, there is a dilemma between approximation accuracy and networks complexity when we use the aforementioned two neural networks to approximate the nonlinear function. However, the Chebyshev neural networks can solve the bottleneck; their hidden layer neuron activation functions are a group of Chebyshev orthogonal polynomials. Nowadays, Chebyshev neural networks are widely used in complex nonlinear systems [1], chaos systems [2], and discrete-time nonlinear systems [3]. Their learning rate and approximation accuracy are better than traditional neural networks, and they can quickly determine the numbers of hidden neurons. Meanwhile, Chebyshev neural networks are widely used in aerospace [4, 5] and chemistry areas [6]. The nanoscale memristor has the potential of information storage because of the nonvolatility with respect to long periods of power-down, so it can be used as synapse in the neural networks. Many researchers in the field of memristors have suggested that this device has high potential for implementing artificial synapses. Afifi et al. studied the realization of STDP learning rules based on the pulsing neuromorphic networks with memristor cross array [7]. Hu et al. built a novel chaotic neural work with memristor to implement associative memory [8]. Wang et al. proposed a PID controller based on memristive CMAC network [9]. Pershin and di Ventra realized the associative memory based on memristive neural networks [10]. Sharifi and Banadaki applied the memristors as memory units to the nonvolatile RAMs and as synapse to the artificial neural networks [11]. Chabi and Klein put forward a neural network with high fault tolerance based on a structure of crossbar switches [12]. Cantley et al. built neural networks with synapses made up from amorphous silicon thin film transistor and memristor, which realized the Hebb learning rules [13]. Gao et al. designed cellular neural network, which is applied to image denoising and edge detection [14]. Kim et al. changed the synapse weight of artificial neural networks with a pulse-based programmable memristor [15].

The memristive Chebyshev neural networks have the following advantages: (a) single-input memristive Chebyshev

neural networks can realize any nonlinear function approximation by only three layers; (b) the number of hidden neurons in memristive Chebyshev neural networks is significantly smaller than that in traditional BP neural networks; (c) we only need to adjust the weight from the hidden layer to the output layer, which can greatly help in quickening the convergence of the algorithm; (d) as memristive synapse is small and passive, the hardware circuits of the memristive Chebyshev neural networks can be implemented by the VLSI circuits easily.

In this paper, we establish the correspondence between memristor conductance and synapse weight through theoretical analysis and MATLAB simulations. The nanoscale memristor is the synapse in Chebyshev neural networks to realize function approximation by memristive BP algorithm and memristive deriving algorithm, respectively.

2. Memristive Synapse

The physical model of the memristor [16] is shown in Figure 1. $W(t)$ and D represent the thickness of the doped layer and total oxide films, respectively. R_{ON} is the value of the memristor when $W(t)$ is equal to D, and R_{OFF} is the value of the memristor when $W(t)$ is equal to zero. $M(0)$ is the initial value of the memristor.

According to the mathematical model of HP memristor, and the flux-controlled model of memristor, we can obtain the following equation [17]:

$$M(t) = \begin{cases} R_{\mathrm{OFF}}, & \varphi(t) < c_1, \\ \sqrt{2k\varphi(t) + M^2(0)}, & c_1 \le \varphi(t) < c_2, \\ R_{\mathrm{ON}}, & \varphi(t) \ge c_2, \end{cases} \tag{1}$$

where $k = (R_{\mathrm{ON}} - R_{\mathrm{OFF}})\mu_v R_{\mathrm{ON}}/D^2$, $c_1 = (R_{\mathrm{OFF}}^2 - M^2(0))/2k$, $c_2 = (R_{\mathrm{ON}}^2 - M^2(0))/2k$. μ_v is the average drift rate of oxygen vacancies, so we can obtain the following equation from (1):

$$G(t) = \frac{1}{M(t)} = \begin{cases} \dfrac{1}{R_{\mathrm{OFF}}}, & \varphi(t) < c_1, \\ \dfrac{1}{\sqrt{2k\varphi(t) + M^2(0)}}, & c_1 \le \varphi(t) < c_2, \\ \dfrac{1}{R_{\mathrm{ON}}}, & \varphi(t) \ge c_2. \end{cases} \tag{2}$$

The derivate of equation (2) is

$$\frac{dG(t)}{dt} = \begin{cases} \left[2k\varphi(t) + M^2(0)\right]^{(-3/2)} \\ \quad \times (-k) \times \dfrac{d\varphi(t)}{dt}, & c_1 \le \varphi(t) \le c_2, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

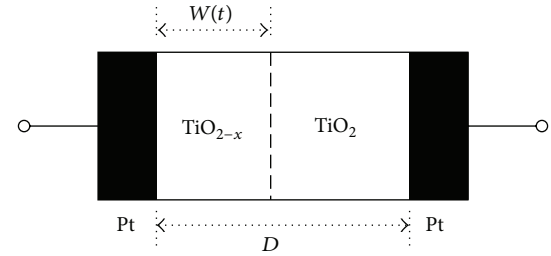


Figure 1: The physical model of the memristor of HP [16].

Using calculus, $dG(t)$ is approximately equal to ΔG when Δt approaches to zero, and then we can obtain the following equation:

$$\Delta G(t) = \begin{cases} \left[2k\varphi(t) + M^2(0)\right]^{(-3/2)} \\ \quad \times (-k) \times \dfrac{d\varphi(t)}{dt} \times \Delta t, & c_1 \le \varphi(t) \le c_2, \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where $d\varphi/dt = v(t)$; the relationship curve between memristive conductance change and voltage is shown in Figure 2. It can be seen that memristive conductance change ΔG increases along with the increasing of applied voltage v. If the learning error e of neural networks is regarded as voltage v, the memristive conductance change can reasonably be described as synapse weight update whose correspondence can be built.

3. Modeling of the Memristive Chebyshev Neural Networks

The nth power orthogonal polynomials $T_n(x)$ which are related to the weight function $\rho(x) = 1/\sqrt{1-x^2}$ are called Chebyshev polynomials of the first kind [18]. Based on definition, if the polynomial system $\{T_n(x)\}$ (where $n = 0, 1, 2, \ldots$) and the weight function satisfy the following relationship:

$$\int_{-1}^{1} \rho(x)\, T_l(x)\, T_k(x)\, dx = \begin{cases} 0, & l \ne k, \\ \displaystyle\int_{-1}^{1} \rho(x)\, T_k^2(x) > 0, & l = k, \end{cases} \tag{5}$$

the orthogonal polynomial system $\{T_n(x)\}$ can be structured in the interval of $[-1, 1]$.

In this paper, a model (see Figure 3) of the single-input memristive Chebyshev neural networks is proposed, which consists of the input layer, hidden layer, and output layer. The weights from input layer to hidden layer are set to 1, and the memristor conductance stores the weights of hidden layer to output layer. The thresholds of all neurons are set to 0. The hidden layer has m neurons, and their transfer functions are a group of Chebyshev orthogonal polynomial functions $T_n(x)$.

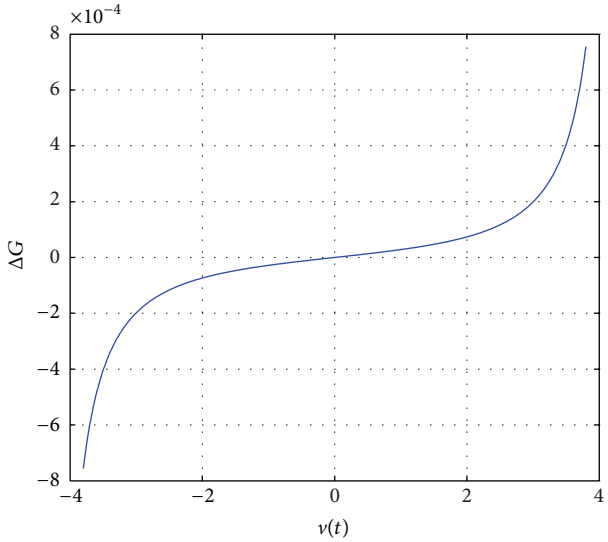


Figure 2: The relation curve of the change of memristive conductance and voltage.

Namely, the hidden layer of the nth neuron transfer function is $T_{n-1}(x)$, where

$$T_0(x) = 1,$$
$$T_1(x) = x, \tag{6}$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n = 1, 2, \ldots$$

4. Realization of Function Approximation Algorithm

Set memristive Chebyshev neural networks model as follows:

input layer: $o = x$;

input of hidden layer neuron: $\text{net}_i = o$;

output of hidden layer neuron: $o_i = T_i(\text{net}_i)$;

output layer:

$$y = \sum_{i=0}^{m-1} W_i O_i = \sum_{i=0}^{m-1} W_i T_i(x). \tag{7}$$

Samples $(x_t, f(x_t))$, $t = 1, 2, \ldots, s$ (s is the number of the samples), are assumed as inputs, and the errors of network output y and the target value $f(x)$ are $e_t = f(x_t) - y_t$.

Network training index is

$$E = \frac{1}{2}\sum_{t=1}^{s} e_t^2. \tag{8}$$

4.1. Memristive BP Algorithm. In order to establish the relationship between the memristor and the synapse, we propose the learning rule for memristive synaptic weight update. According to (4), we assume the memristive BP algorithm as follows:

$$\Delta W_i = \left[2k\Gamma(t) + M^2(0)\right]^{(-3/2)} \times (-k) \times e_t(t) \times \Delta t \times T_i(x_t),$$
$$W_i(K+1) = W_i(K) + \Delta W_i(K), \tag{9}$$

where $i = 0, 1, 2, \ldots, m-1$, $\Delta t = \eta$ is the learning rate, K is the number of learning, and Γ is the integral of the error e.

The learning algorithm is described as follows.

Step 1. Take any number of hidden neurons $n \geq 3$, choose the initial weights $W_i(0)$ as initial memristor conductance, let the learning rate $0 < \eta < 1$, give the small positive number ε, and set the training sample set $(x_t, f(x_t))$ as $E = 0$, $t = 1$, $K = 0$.

Step 2. Calculate

$$y_t(K) = \sum_{i=0}^{m-1} W_i T_i(x_t),$$
$$e_t = f(x_t) - y_t(K), \qquad E \longleftarrow E + 0.5e_t^2. \tag{10}$$

Step 3. Adjust weight,

$$W_i(K+1) = W_i(K) + \Delta W_i(K). \tag{11}$$

Step 4. $t \leftarrow t + 1$; if $t < s$, then jump back to Step 2, otherwise Step 5.

Step 5. If $E \leq \varepsilon$, then stop the learning; otherwise, $E = 0$, $t = 1$, $K \leftarrow K + 1$, and jump back to Step 2.

4.2. Memristive Deriving Algorithm. The basic principles of the memristive deriving algorithm are as follows. Take any less n hidden neurons as the initial cells. After training K times, the error e *is* no longer changed; that is, $E(K) = E(K-1) > \varepsilon$ (ε is a pretraining given precision index). The network derives automatically by adding a hidden neuron ($n \leftarrow n+1$). The previous process is repeated until $E(K) \leq \varepsilon$ when the hidden neurons stop deriving and learning. Until reaching target accuracy, the neural network topology is automatically generated.

The specific algorithm can be described as follows.

Step 1. Take number of hidden neurons $n = 2$, choose the initial weights $W_{ij}(0)$ and $C_j(0)$ as initial memristor conductance, let the learning rate $0 < \eta < 1$, give the small positive number $\varepsilon > 0$, and set the training set $\{(x_t, d_t), t = 1, 2, \ldots, s\}$ and the number of learning $K = 1$.

Step 2. Input of hidden layer neuron is

$$\text{net}_j = \sum_{i=1}^{m} W_{ij} x_i, \quad j = 0, 1, 2, \ldots, n-1. \tag{12}$$

Calculate

$$y_t(K) = \sum_{j=0}^{n} C_j(K-1)\, T_j(\text{net}_j), \qquad e_t = d_t - y_t. \tag{13}$$

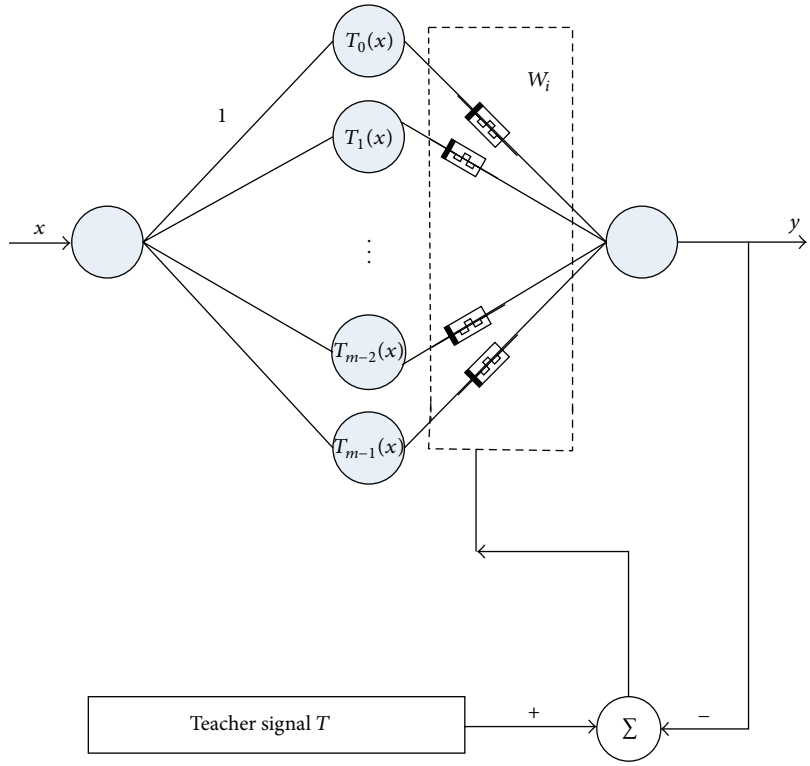


Figure 3: Model of single-input memristive Chebyshev neural networks.

Step 3. Weight update rule is

$$\begin{aligned}
\Delta C_j(K) &= \left[2k\Gamma(t) + M^2(0)\right]^{(-3/2)} \\
&\quad \times (-k) \times e_t(t) \times \Delta t \times T_i\left(\text{net}_j\right), \\
C_j(K) &= C_j(K-1) + \Delta C_j(K), \\
\Delta W_{ij}(K) &= \left[2k\Gamma(t) + M^2(0)\right]^{(-3/2)} \\
&\quad \times (-k) \times e_t(t) \times C_j \Delta t \times T_j'\left(\text{net}_j\right) \times x_i, \\
W_{ij}(K+1) &= W_{ij}(K-1) + \Delta W_{ij}(K).
\end{aligned} \tag{14}$$

Step 4. Calculate

$$E(K) = \frac{1}{2}\sum_{t=1}^{s} e_t^2. \tag{15}$$

If $E(K) \le \varepsilon$, stop derivative and learning which illustrate that neural network topology is automatically generated. Otherwise, go to Step 5.

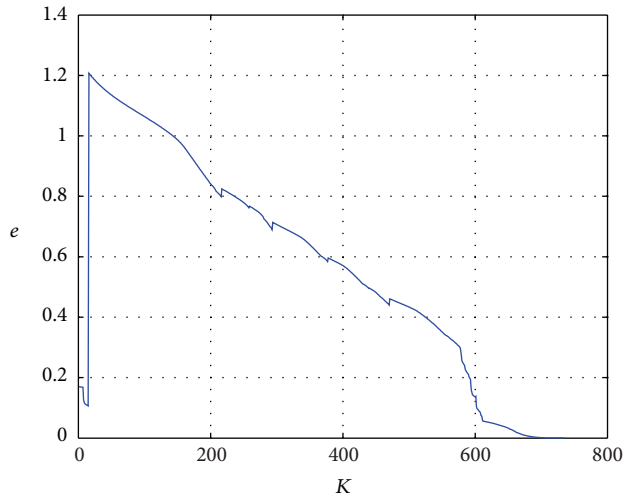
Step 5. If $E(K) = E(K-1)$, the training error of the network has no longer reduced. We should increase the number of hidden neurons in order to improve network performance. Namely, $n \leftarrow n+1$, $K \leftarrow K+1$, and jump to Step 2 to continue learning.

5. Experimental Results

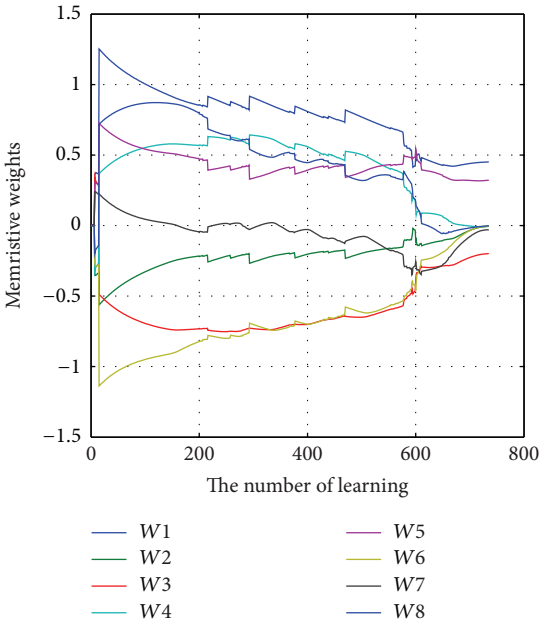
We use the memristive Chebyshev neural networks to realize nonlinear function $y = x^3 \sin x + \sin^2 x + \cos(3x)$ approximation through memristive BP algorithm and deriving algorithm, respectively. We suppose that memristor model parameters are $R_{\text{ON}} = 100\,\Omega$, $R_{\text{OFF}} = 10000\,\Omega$, $M(0) = 1000\,\Omega$, $D = 10\,\text{nm}$, $\mu_v = 10^{-14}\,\text{m}^2\,\text{s}^{-1}\,\text{V}^{-1}$, $G(0) = 1/M(0)$.

5.1. Memristive BP Algorithm Realizes Function Approximation. The $1 \times 8 \times 1$ network structure is shown in Figure 3. The learning rate $\eta = 0.02$. The number of samples $s = 200$. The number of learning is 1000. The target mean square error $E = 1.0 \times 10^{-5}$. The experimental result is shown in Figure 4. The relationship between training error and the number of learning is shown in Figure 4(a). The number of final learning is 735 when the minimum training error meets the requirements of function approximation. The memristive synaptic weights are constantly updated in the learning process as shown in Figure 4(b). Figures 4(c) and 4(d) describe the final memristive weights and the approximation curve of actual output signal relative to the teacher signal, respectively.

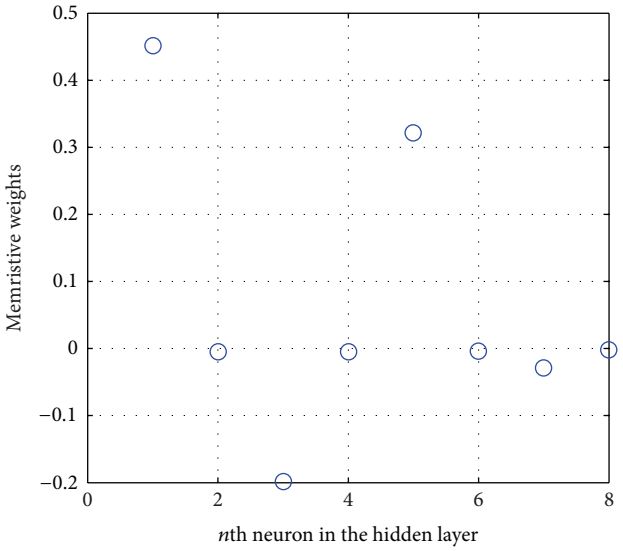
5.2. Memristive Deriving Algorithm Realizes Function Approximation. We adopt the $1 \times 2 \times 1$ initial neural network, with $\varepsilon = 1.0 \times 10^{-5}$ and learning rate 0.02, and the experimental result is shown in Figure 5. The final network structure is $1 \times 25 \times 1$, and the time of learning is 420. Figure 5(a) shows the final memristive synaptic weights in hidden layer. Figure 5(b)



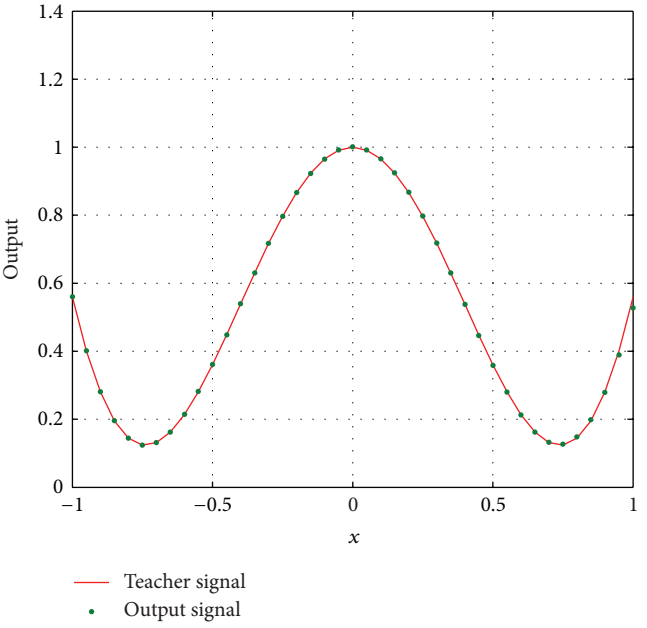
(a) Relationship between training error and the number of learning



(b) Relationship between memristive weights and the number of learning



(c) The final memristive weights in the hidden layer



(d) The approximation curve

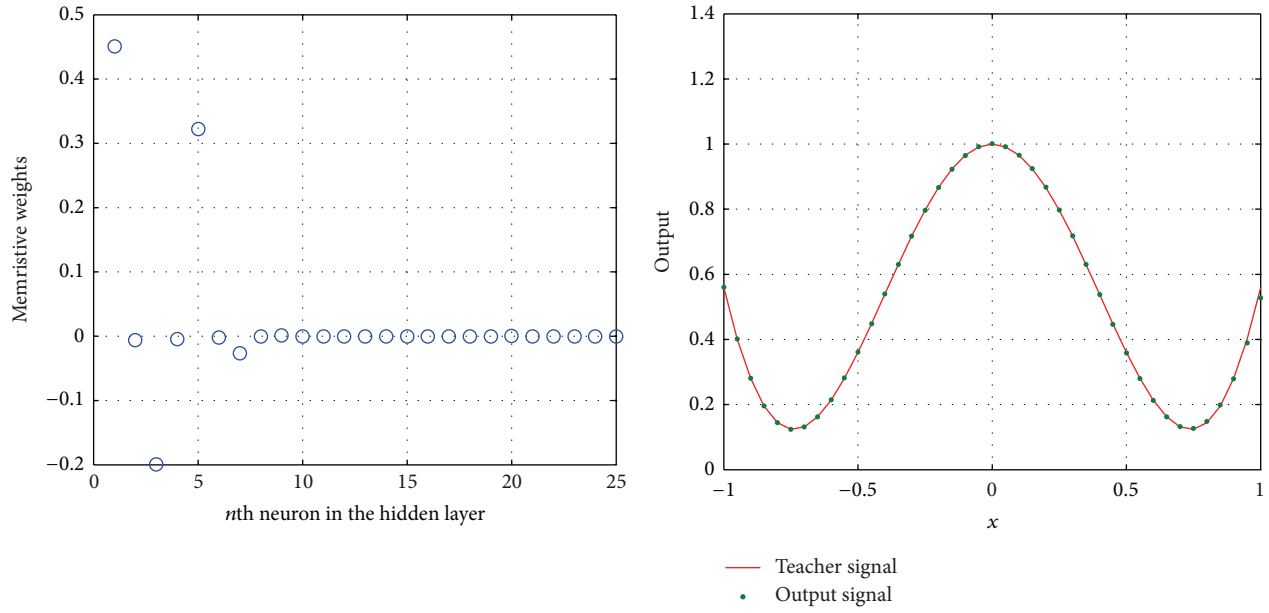
Figure 4: Simulation results for memristive BP algorithm.

shows the approximation curve of the actual output signal relative to the teacher signal.

Compared with the memristive BP algorithm, the memristive deriving algorithm has less learning times (735 – 420 = 315) for its neural network is automatically generated.

6. Conclusions

The HP memristor model is analyzed, and the correspondence between the memristive conductance change and the synapse weight update in Chebyshev neural networks is established. Two learning algorithms of memristive synapse weight update are proposed. The nonlinear function approximation is implemented by memristive BP algorithm and memristive deriving algorithm, respectively. On the one hand, the memristive BP networks have fixed structure which is easily implemented. On the other hand, the memristive derivative networks have dynamic structure reaching the optimum solution. Memristive Chebyshev neural networks are hopeful to be used in aspects of pattern recognition and data compression.



(a) The final memristive weights in the hidden layer

(b) The approximation curve

Figure 5: Simulation result for memristive deriving algorithm.

Acknowledgments

The work was supported by National Natural Science Foundation of China (Grant nos. 60972155 and 61101233), Fundamental Research Funds for the Central Universities (Grant nos. XDJK2012A007 and XDJK2013B011), University Excellent Talents Supporting Foundations of Chongqing (Grant no. 2011-65), University Key Teacher Supporting Foundations of Chongqing (Grant no. 2011-65), Technology Foundation for Selected Overseas Chinese Scholars, Ministry of Personnel in China (Grant no. 2012-186), National Science Foundation for Postdoctoral Scientists of China (Grant no. CPSF20100470116), and "Spring Sunshine Plan" Research Project of Ministry of Education of China (Grant no. z2011148).

References

[1] S. Purwar, I. N. Kar, and A. N. Jha, "On-line system identification of complex systems using Chebyshev neural networks," *Applied Soft Computing Journal*, vol. 7, no. 1, pp. 364–372, 2007.

[2] P. Akritas, I. Antoniou, and V. V. Ivanov, "Identification and prediction of discrete chaotic maps applying a Chebyshev neural network," *Chaos, Solitons and Fractals*, vol. 11, no. 1, pp. 337–344, 2000.

[3] A. K. Shrivastava and S. Purwar, "State feedback and output feedback tracking control of discrete-time nonlinear system using Chebyshev neural networks," in *Proceedings of the International Conference on Power, Control and Embedded Systems (ICPCES '10)*, pp. 1–6, Allahabad, India, December 2010.

[4] A. M. Zou, K. D. Kumar, and Z. G. Hou, "Quaternion-based adaptive output feedback attitude control of spacecraft using chebyshev neural networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 9, pp. 1457–1471, 2010.

[5] A. M. Zou, K. D. Kumar, Z. G. Hou, and X. Liu, "Finite-time attitude tracking control for spacecraft using terminal sliding mode and chebyshev neural network," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 41, no. 4, pp. 950–963, 2011.

[6] S. P. Yan, M. F. Peng, J. F. Lei et al., "CO_2 concentration detection based on Chebyshev neural network and best approximation theory," *Instrument Technique and Sensor*, vol. 6, pp. 107–110, 2011.

[7] A. Afifi, A. Ayatollahi, and F. Raissi, "Implementation of biologically plausible spiking neural network models on the memristor crossbar-based CMOS/nano circuits," in *Proceedings of the European Conference on Circuit Theory and Design Conference Program (ECCTD '09)*, pp. 563–566, Antalya, Turkey, August 2009.

[8] X. F. Hu, S. K. Duan, and L. D. Wang, "A novel chaotic neural network using memristors with applications in associative memory," *Abstract and Applied Analysis*, vol. 2012, Article ID 405739, 19 pages, 2012.

[9] L. D. Wang, X. Y. Fang, S. K. Duan et al., "PID controller based on memristive CMAC network," *Abstract and Applied Analysis*, vol. 2013, Article ID 510238, 2013.

[10] Y. V. Pershin and M. di Ventra, "Experimental demonstration of associative memory with memristive neural networks," *Neural Networks*, vol. 23, no. 7, pp. 881–886, 2010.

[11] M. J. Sharifi and Y. M. Banadaki, "General spice models for memristor and application to circuit simulation of memristor-based synapses and memory cells," *Journal of Circuits, Systems and Computers*, vol. 19, no. 2, pp. 407–424, 2010.

[12] D. Chabi and J. O. Klein, "Hight fault tolerance in neural crossbar," in *Proceedings of the 5th Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS '10)*, pp. 1–6, Hammamet, Tunisia, March 2010.

[13] D. K. Cantley, A. Subramaniam, H. J. Stiegler et al., "Hebbian learning in spiking neural networks with nanocrystalline silicon

TFTs and memristive synapses," *IEEE Transactions on Nanotechnology*, vol. 10, no. 5, pp. 1066–1073, 2011.

[14] S. Y. Gao, S. K. Duan, and L. D. Wang, "On memristive cellular neural network and its applications in noise removal and edge extraction," *Journal of Southwest University*, vol. 33, no. 11, pp. 63–70, 2011.

[15] H. Kim, M. P. Sah, C. J. Yang, T. Roska, and L. O. Chua, "Neural synaptic weighting with a pulse-based memristor circuit," *IEEE Transactions on Circuits and Systems I*, vol. 59, no. 1, pp. 148–158, 2012.

[16] D. B. Strukov, G. S. Snider, D. R. Stewart et al., "The missing memristor found," *Nature*, vol. 453, pp. 80–83, 2008.

[17] L. D. Wang, E. Drakakis, S. K. Duan et al., "Memristor model and its application for chaos generation," *International Journal of Bifurcation and Chaos*, vol. 22, no. 8, Article ID 1250205, 14 pages, 2012.

[18] G. D. Mo and K. D. Liu, *Function Approximation Methods*, Science Press, Beijing, China, 2003.