2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

A Spiking-based Mechanism for Self-organizing RBF Neural Networks

Honggui Han, Lidan Wang, Junfei Qiao, and Gang Feng

Abstract— **In this paper, a spiking growing algorithm (SGA) is proposed for optimizing the structure of radial basis function (RBF) neural network. Inspired by the synchronous behavior of spiking neurons, the spiking strength (*ss*) of the hidden neurons is defined as the criteria of SGA, which investigates a new way to simulate the connections between hidden and output neurons of RBF neural network. This SGA-based RBF (SGA-RBF) neural network can self-organize the hidden neurons online, to achieve the appropriate network efficiency. Meanwhile, to ensure the accuracy of SGA-RBF neural network, the structure-adjusting and parameters-training phases are performed simultaneously. Simulation results demonstrate that the proposed method can obtain a higher precision in comparison with some other existing methods.**

Index Terms—**Spiking-based mechanism, spiking-based growing algorithm, self-organizing radial basis function neural network, nonlinear system.**

I. Introduction

BEING inherited from the concept of biological receptive fields, radial basis function (RBF) neural network has been proposed. Meanwhile, due to the simple structure and universal approximation ability, RBF neural network has been widely used for nonlinear system modeling, adaptive control, and fault diagnosis [1]–[3]. However, most of these studies fixed the number of hidden neurons [4]. Thus there are two main disadvantages: if the number of hidden neurons is too large, heavy computational burden and over-fitting phenomenon will be occurred. On the contrary, if the number of hidden neurons is too small, it may likely to with a result of low precision [5]. One of the challenges for RBF neural network is to self-organize the structure to improve the approximation accuracy.

In order to design a proper size of RBF neural network, some methods and strategies have been proposed to realize the self-organizing adjustment of RBF structure [6]. Chen *et al.* proposed an orthogonal least squares (OLS) algorithm to add the hidden neurons one by one until catching the required network [7]. However, the OLS algorithm has a heavy computing burden which calculated by the process of Gram-Schmidt orthogonalization, especially for large training data set [8]. To overcome this problem, some improved OLS algorithms have been proposed in [9]–[11]. These algorithms can improve the training speed to organize the structure of RBF neural network. But, all these algorithms are dependent on the total training samples, this means that the OLS-based methods are primarily used for offline learning [12].

Recently, some sensitivity analysis (SA) methods have been proposed to design the structure of RBF neural network. Shi *et al.* proposed a SA algorithm for constructing the structure of RBF neural network [12]. The number of hidden neurons can be calculated by the maximization of the output's sensitivity to the training data. However, it is difficult to own the geometric probability density functions of the input variables [13]. To decompose the variance of the output as a sum of contributions of each input variable, Saltelli *et al.* used the Fourier transform sensitivity method to decompose the signal function into an infinite number of different frequency sine signals [14]. Han *et al.* introduced a Fourier transform SA algorithm to design a dynamic RBF neural network [15]. The simulation results show that the network performances, such as approximation accuracy, are improved. However, ill-conditioned problems may occur because of their stochastic nature.

Resource allocation network (RAN) [16] was designed by Platt and RANEK [17], which trained with extended Kalman filter (EKF), was developed and improved based on the RAN. To avoiding the scale of RBF become unnecessarily large, the minimal resources allocating network (MRAN) proposed with the concept of the contribution of neuron to the overall network output [18]–[19]. However, various parameters and thresholds in pruning process impede its application for solving complex nonlinear problems. Huang *et al.* proposed the growing and pruning RBF (GAP-RBF) [20], and generalized growing and pruning RBF (GGAP-RBF) [21] which relies on the concept of neuron's significance. Though these algorithms generate RBF models in terms of compactness and generalization ability, the significance relies

H. G. Han is with College of Electronic and Control Engineering, Beijing University of Technology, Beijing, and also with the Department of Mechanical and Biomedical Engineering, City University of Hong Kong, Kowloon, Hong Kong. (Corresponding author to provide phone: 010-67391631; e-mail: rechardhan@sina.com).

L. D. Wang studies at College of Electronic and Control Engineering, Beijing University of Technology, Beijing, China. (e-mail: wanglidan01@163.com).

J. F. Qiao is with College of Electronic and Control Engineering, Beijing University of Technology, Beijing, China. (e-mail: isibox@sina.com).

G. Feng is with the Department of Mechanical and Biomedical Engineering, City University of Hong Kong, Kowloon, Hong Kong, and also with the Nanjing University of Science and Technology, Nanjing 210094, China(e-mail: megfeng@cityu.edu.hk).

This work was supported by the National Science Foundation of China under Grant 61203099; National Science Foundation of China under Grant 61034008; Ph.D. Programs Foundation of Ministry of Education of China under Grant 20121103120020; Beijing Municipal Natural Science Foundation under Grant 4122006; Beijing Nova Program under Grant Z131104000413007; HongKong Scholor Program under Grant XJ2013018.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

on all the input information.

Moreover, some optimization methods have been employed to organize the RBF neural network [22]. Ferentions *et al.* proposed a genetic algorithm which is a biological plausible to reduce the hidden neurons [23]. Yao *et al.* and Alex *et al.* respectively proposed particle swarm optimization algorithms to train the structure and parameters of RBF neural network [24]–[25]. These algorithms can realize the construction of RBF neural network. However, these optimization methods are computation expensive, especially when the search space is huge.

Different from above algorithms, this paper constructs the RBF neural network via mimic biological neurons systems and information processing ability of the human brain [26]–[27]. A novel structure growing approach (SGA) is proposed to design the RBF neural networks. The structure-adjusting phase and parameters-training phase are performed simultaneously for the SGA-RBF neural network.

The rest of the paper is organized as follows. Section 2 describes the RBF neural network briefly. The concrete spiking-based structure adjusting mechanism and parameters training methods are given in Section 3. In Section 4, the computation complexity of proposed SGA-RBF neural network is discussed and compared with the other methods. In Section 5, function approximation experiments and nonlinear system modeling simulation are studied to verifying the effectiveness of the proposed algorithm, and the comparison with several other approaches presented. Finally, this paper concludes with a discussion of the further development in Section 6.

I. Structure of Radial Basis Function (RBF) Neural Network

The typical RBF neural network has a three-layer forward structure. The first layer is input layer, which passes the input data to the hidden layer. The neurons in the hidden layer simultaneously receive and process the information from the input layer, each neuron in this layer has a Gaussian activation function. The third layer is a linear calculation called output layer.

A multi-input-single-output (MISO) RBF neural network is shown in Fig.1. The input vector is $\mathbf{x}=[x_1, x_2, \cdots, x_i]$, the total number of hidden neurons is J. For the pth sample, the I-dimensional input values are mapped into the J-dimensional outputs by the function $\boldsymbol{x}$

$$\boldsymbol{\Phi}_j\left(\mathbf{x}_p\right)=\exp\left(-\frac{\left\|\mathbf{x}_p-\mathbf{c}_j\right\|^2}{2\sigma_j^2}\right), \tag{1}$$

where $\|\cdot\|$ represents the Euclidean distance between vector $\mathbf{x}_p$ and $\mathbf{c}_j$, $j=1,2,\ldots,J$, $\Phi_j(\cdot)$ is the activation function of the jth hidden neuron, $\mathbf{c}_j$ and σ_j represent the center and width of the jth hidden neuron respectively.

For the pth sample, the network output is defined as

$$o_p=\sum_{j=1}^{J} w_j \boldsymbol{\Phi}_j\left(\mathbf{x}_p\right), \tag{2}$$

where w_j is the weight connection between the jth hidden unit and output neuron.

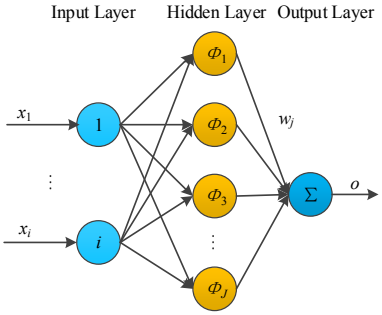

Fig. 1. The structure of RBF neural network

II. The Spiking-Growing Algorithm-based RBF (SGA-RBF) Neural Network

Two parts are given in this section: the SGA mechanism for structure design and the gradient learning approach for parameters adjusting.

A. SGA Mechanism

In [27], the Leaky Integrate-and-Fire (LIF) model of biological neurons is expressed as:

$$\tau_m \frac{dV}{dt}=-V(t)+RI(t), \tag{3}$$

where $V(t)$ is the membrane potential of biological neuron, τ_m is the membrane time constant (describe the time span that the action potential decreases to 0), R is the membrane resistance and $I(t)$ is an injected current. The postsynaptic voltage varies with time as shown in Fig.2.

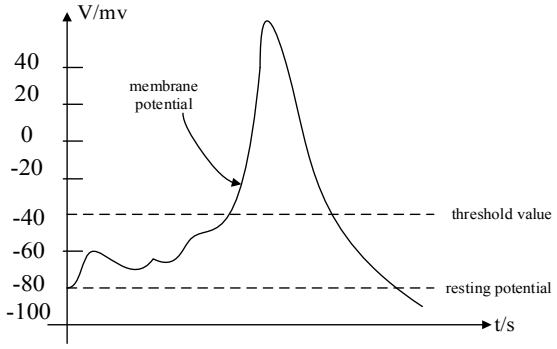

Fig. 2. The postsynaptic voltage over time

In this paper, the time scales of neurons' duty-cycle are given as [28]

$$\tau_m(V)=\tau_2+\frac{\tau_1-\tau_2}{1+\exp(-\frac{V}{k_\tau})}, \tag{4}$$

where τ_1 and τ_2 denotes different time scales. k_τ is a sign that characterize the slope of synaptic activation.

Define $\tau_2=0$, $k=\tau_1$, based on the connection mode between different cortex, an function for spiking strength (ss), is designed as

$$ss_j=-k_\tau \ln(\frac{k}{\sin(\boldsymbol{\Phi}_j)+\varepsilon}-1), \tag{5}$$

where both k and k_τ are constants, Φ_j is the output value of the jth hidden neuron. The **sine** function is introduced into the spiking strength function due to the characteristic of duty period in spiking neurons, and ε is a small positive number for the purpose of avoiding the calculation trouble when $sin(\Phi_j)=0$.

In biological spiking neurons, the membrane potential value indicates the excitement level at different times. The relationship between the spiking strength (ss) and the outputs of hidden neuron (Φ_j) in (5) is clearly depicted in Fig.3. The curve presents periodically spiking.

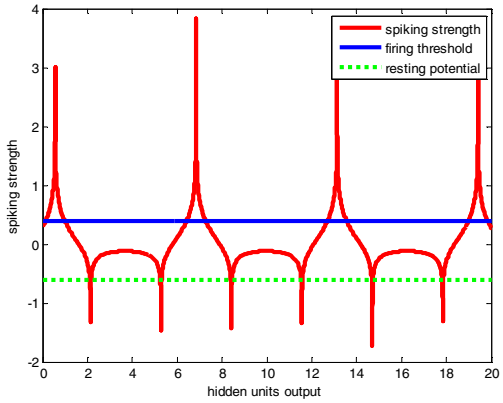

Fig. 3. The curve between ss and hidden unit output

If the value of ss_j is larger than the firing threshold ss_0 and the curve is on a growth trend, the neuron is an excited one and will be splitted. When the value of ss_j is less than the resting potential, the jth neuron is inactive one and will be deleted to obtain a parsimonious structure of the network.

According to (5), the derivation of ss can be rewritten as

$$\frac{dss(\Phi_j)}{d\Phi_j}=\frac{kk_\tau\cos(\Phi_j)}{\left[\sin(\Phi_j)+(k-\varepsilon)\right]\left[\sin(\Phi_j)+\varepsilon\right]}. \tag{6}$$

To make the growing mechanism effectively, the concept of squared error percentage E_r is defined as [29]–[30]

$$E_r=100\times\frac{o_{\max}-o_{\min}}{P}\sum_{p=1}^{P}e_p^2, \tag{7}$$

o_{max}, o_{min} represent the maximum value and minimum value of the proposed SGA-RBF output, P is the total number of training samples.

The centers and widths of the new added hidden neurons are expressed as [31]–[32]:

$$\begin{cases}\mathbf{c}_{jm}=\alpha_m\mathbf{c}_j+\beta_m\mathbf{x},\\ \sigma_{jm}=\alpha_m\sigma_j,\\ m=1,2,\cdots,N_{new},\end{cases} \tag{8}$$

where $\mathbf{c}_j$ and σ_j represents the center and variance of pre-split jth neuron, N_{new} is the number of the new added unit, $\mathbf{c}_{jm}$ and σ_{jm} respectively represents the center and variance of the new added mth neuron, $\alpha_m\in[0.95,1.05]$, $\beta_m\in[0,0.1]$.

The output weights of the new added hidden neurons can be defined as [31]–[32]

$$\begin{cases}w_{jm}=\gamma_m\dfrac{w_j\cdot\Phi_j(\mathbf{x}_p)-e}{N_{new}\cdot\Phi_{jm}(\mathbf{x}_p)},\\ \sum_{m=1}^{N_{new}}\gamma_m=1,\\ m=1,2,\cdots,N_{new},\end{cases} \tag{9}$$

where w_{jm} represents the connection weight between the new added neuron m and the output layer, w_j represents the connection weight between the pre-split hidden neuron and the output layer, e is the current error of the network.

B. The Gradient Learning Approach

The parameters were trained contains three parts: the output weight, the center and the width, the performance evaluation function as follows:

$$e_p=y_p-o_p, \tag{10}$$

$$E_p=\frac{1}{2}\sum_{p=1}^{P}(y_p-o_p)^2, \tag{11}$$

where y_p is the desired output and o_p is the actual output. There will be:

$$\frac{\partial E_p}{\partial w_j}=-\frac{\partial o_p}{\partial w_j}\times\sum_{p=1}^{P}e_p, \tag{12}$$

$$\frac{\partial E_p}{\partial \sigma_j}=-\frac{\partial o_p}{\partial \Phi_{j(\mathbf{x}_p)}}\frac{\partial \Phi_{j(\mathbf{x}_p)}}{\partial \sigma_j}\times\sum_{p=1}^{P}e_p, \tag{13}$$

$$\frac{\partial E_p}{\partial c_j}=-\frac{\partial o_p}{\partial \Phi_{j(\mathbf{x}_p)}}\frac{\partial \Phi_{j(\mathbf{x}_p)}}{\partial c_j}\times\sum_{p=1}^{P}e_p. \tag{14}$$

According to the equations (1)-(2) and (10)-(11), the equations (12)-(14) can be rewritten as

$$\frac{\partial E_p}{\partial w_j}=-\sum_{p=1}^{P}e_p\Phi_j(\mathbf{x}_p), \tag{15}$$

$$\frac{\partial E_p}{\partial \sigma_j}=-\frac{\sum_{p=1}^{P}e_p w_j\Phi_j(\mathbf{x}_p)\left\|\mathbf{y}_{p,j}-\mathbf{c}_j\right\|^2}{\sigma_j^3}, \tag{16}$$

$$\frac{\partial E_p}{\partial c_j}=-\frac{\sum_{p=1}^{P}e_p w_j\Phi_j(\mathbf{x}_p)(\mathbf{x}_p-\mathbf{c}_j)}{\sigma_j^2}. \tag{17}$$

Then the gradient learning approach is

$$w_j(\mathbf{x}_{p+1})=w_j(\mathbf{x}_p)-\eta_w\frac{\partial E_p}{\partial w_j}, \tag{18}$$

$$\sigma_j(\mathbf{x}_{p+1})=\sigma_j(\mathbf{x}_p)-\eta_\sigma\frac{\partial E_p}{\partial \sigma_j}, \tag{19}$$

$$c_j(\mathbf{x}_{p+1})=c_j(\mathbf{x}_{p+1})-\eta_c\frac{\partial E_p}{\partial c_j}. \tag{20}$$

Based on the former analysis, the details of proposed SGA-RBF adjustment process is summarized as follows.

Step1: Create an initial RBF neural network with a small number of hidden neurons, the number of neurons in the input

layer equals to the dimensions of the input vector. The number of output layer units depends on the output variables. The center and width of Gaussian function and the output weight is randomly generated.

Step2: The output weights, centers and widths of hidden units are trained by the gradient descent algorithm.

Step3: Calculate the equations (5)-(7), if the three rules are satisfied: ss_j is larger than the threshold ss_0 and its derivative is a positive number, E_r is no less than the expectation value E_0 such as

$$\begin{cases} ss_j > ss_0, \\ \dfrac{dss}{d\Phi_j} > 0, \\ E_r \geq E_0. \end{cases} \tag{21}$$

split the jth neuron, adjust the structure of the RBF neural network and reset the parameters of the new added units based on the equation (8)-(9).

Step4: Train the output weight, the center and width of hidden neurons by the gradient descent algorithm.

Step5: If the training phase meets the precision requirement or reaches the training epochs, stop the iteration. Otherwise go to Step3.

III. Computational Complexity

Both the time computation cost and memory space requirement of the proposed SGA-RBF are discussed in this section.

Table.1 The computational complexity comparison

Computation Cost	SGA-RBF	$Maxepoch \times J$
	OLS	J^3
Memory Requirement	SGA-RBF	$3(1,J)+(I,J)+(P,J)$
	OLS	$4(1,J)+(I,J)+2(P,P)$

For each sample, the complexity level in the training phase depends both on the number of hidden neurons and the iterations. The time complexity of the gradient descent method is O ($J \times iters$) and the structure growing phase is O ($J \times iters$). So the computational complexity in the structure adjusting and parameter training process which performs simultaneously is O ($J \times iters$). Where, J is the number of hidden neurons and the *iters* means the total epochs in the training phase, generally, $J << iters \leqslant Maxepoch$. The OLS method relies on the implementation of the pseudoinverse technique to calculate the weight matrix. The time complexity of the pseudoinverse method is O (J^3), where J depends on the maximum number of hidden neurons, which equals to the number of input samples. If the number of the training samples is slightly larger, the OLS method will need more training time than the SGA algorithm.

The memory space requirement for proposed SGA-RBF is $3(1, J) + (I, J) + (P, J)$, the space occupation for the centers of hidden neurons is (I, J), for the outputs of hidden neurons is (P, J), for the radius of hidden neurons, the weight values and ss values of hidden neurons are $(1, J)$. The memory space occupation for the traditional OLS method is $4(1, J) + (I, J) + 2(P, P)$, including the space of centers (I, J), radius $(1, J)$, weight vector $(1, J)$, output of hidden neuron $(1, J)$ and the inverse weight vector $(1, J)$. In addition, the space requirement for Gram-Schmidt orthogonalization process is $2(P, P)$, P is the total number of training samples. From the above calculation results, the space complexity of OLS RBF neural network is larger than the proposed SGA-RBF neural network.

IV. Simulations

To demonstrate the effectiveness and applicability of the proposed SGA-RBF neural network, three experiments are discussed.

The root mean square error is used to evaluate the training procedure in the experiments.

$$\text{RMSE} = \sqrt{\sum_{p=1}^{P} e_p^{\,2} \Big/ P}, \tag{22}$$

where p is the number of samples, P is the total number of training samples.

A. Nonlinear Hermit function

The first experiment is designed to approximate the nonlinear Hermit polynomial function, which was proposed in [33], by the proposed SGA-RBF neural network.

$$y = 1.1(1 - x + 2x^2)\exp(-\frac{x^2}{2}), \tag{23}$$

the variable x satisfies the uniform distribution $U \in [-4, 4]$. For each trail, the size of training samples is 100, the size of testing samples is 101. The initial number of hidden neurons is 2, the output weights and hidden function parameters are randomly generated.

For the sample p the output of RBF neural network is o_p, the desired output is y_p, the error is calculated by (11). The max training epochs is 200. The experimental results are shown in Figs. 4-6 and Table1.

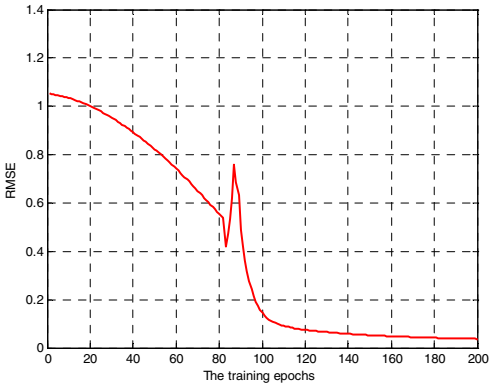

Fig. 4. The training error

Fig. 4 shows the RMSE values in the training process within 200 steps. The error values present oscillation when new neurons are added to the hidden layer. However, the RMSE values decrease faster after the structure organization. Fig. 5 gives the concrete dynamic adjustment form of the hidden neurons in the training process. The results show that the proposed spiking-based mechanism can self-organize the network structure.

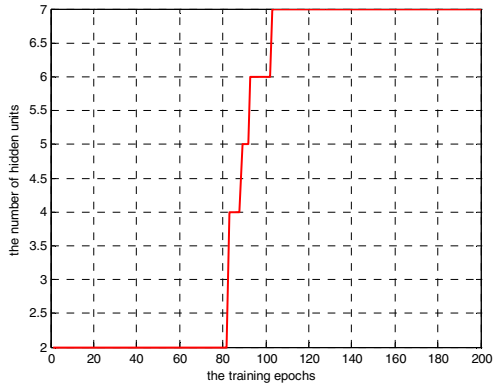

Fig. 5 The number of hidden neurons

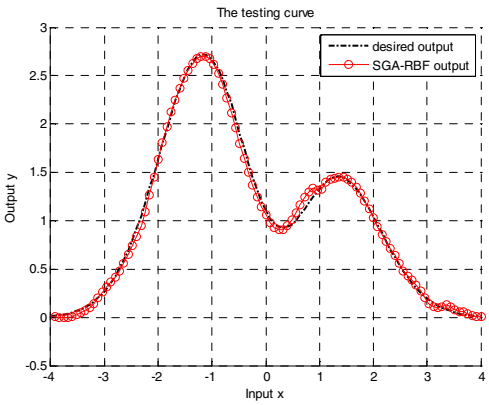

Fig. 6. The testing sample output

Table.2 The performance comparison of different algorithms

Algorithms	*Hidden neurons*	*Running time(s)*	*RMSE*	
			training	*testing*
SGA-RBF	**7**	**0.65**	**0.0366**	**0.0012**
RBF	10	9.46	0.0889	0.0028
RAN[16]	14*	1.12*	0.0433*	0.0073*
OLS[34]	6	0.80*	0.4489*	0.5050
BIC-OLS[34]	3	*	*	0.1793

*There are no results listed in the original papers.
The number of RBF algorithm was dynamically selected from the consecutive integer 1 to 30. The RBF neural network with 10 hidden neurons has the best training and testing RMSE values.

In Fig. 6, the outputs of the SGA-RBF neural network are presented to compare with real values. The testing outputs of the proposed SGA-RBF neural network close to the desired outputs. The performance comparison with other algorithms is presented in Table 2.

Based on Table 2, after 200 training process, the number of hidden neurons in proposed SGA-RBF neural network is smaller than the traditional RBF neural network and Resource allocating network (RAN). Meanwhile, the proposed SGA-RBF neural network is more accuracy than the two algorithms. By comparing with the Orthogonal least squares (OLS) method [34] and Bayesian information criterion (BIC-OLS) method [34], the number of hidden neurons in proposed SGA algorithm is larger and the testing RMSE value is far less than the two methods. Comprehensive comparison among all algorithms presented in Table 2, the training RMSE value is the smallest and the training time is the shortest in the first four methods. Moreover, the testing RMSE value is the minimum one in all listed algorithms. The highest testing accuracy means that SGA-RBF owns the best generalization ability in all listed methods. A higher precision with a more parsimonious structure, shorter training time, it indirectly suggests that the proposed SGA mechanism is efficient to design the RBF neural network by plausible biological mode.

Table 2 demonstrates that the training time of the proposed SGA-RBF is shorter than that of OLS method. This result manifests that the SGA algorithm is efficient to construct the RBF neural network.

*B. 2-D **sine** function*

Approximate the two-dimensional function [31]

$$z = \frac{\sin(x)\sin(y)}{x \cdot y} \tag{24}$$

There are two inputs (x, y) and one output z in the proposed SGA-RBF neural network. The variables x and y are both generated randomly within [-1, 1]. The size of training samples and testing samples are both 400. The hidden output weights and hidden activation function parameters are randomly generated within a small scope. The initial number of hidden neurons is 3. The pre-set training error was 0.01. The experimental results are shown as Figs. 7-9.

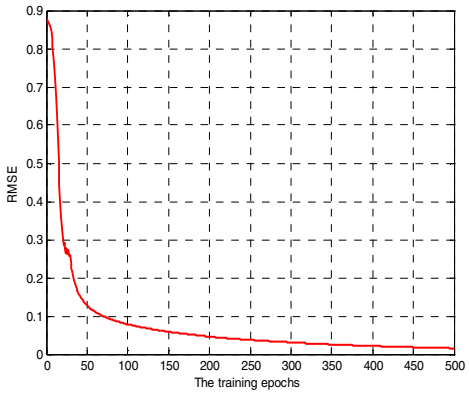

Fig. 7. The training root-mean-square-error

Fig. 7 indicates the RMSE values in the training process within 500 steps. The RMSE values are also with slight oscillation when new neurons are inserted to the hidden layer. And the RMSE values keep a state of decrease quickly when the structure is adjusted. Fig. 8 gives the dynamic details of the hidden neurons in the training process. The testing outputs of SGA-RBF neural network is compared in Fig. 9. It can be seen that the testing values of the proposed SGA-RBF neural network are close to the real points.

Fig. 8 shows that the ultimate number of hidden neurons is 12 after the training process. The testing RMSE value is 0.0021. The traditional RBF neural network were experimented to compare with the proposed SGA-RBF neural

network. The number of hidden neurons in the RBF neural network was chosen from consecutive integer 1 to 40. In the dynamic selection process, the RBF neural network, which has 15 neurons in the hidden layer, obtains the smallest training RMSE value. The testing RMSE value of the RBF neural network is 0.0099. It can be seen that proposed SGA-RBF obtains a smaller testing RMSE value using fewer hidden neurons. The results suggest that the proposed SGA method can self-organize the network structure.

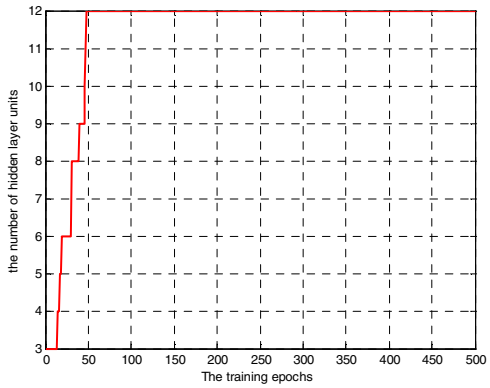

Fig. 8. The number of hidden units in the training phase

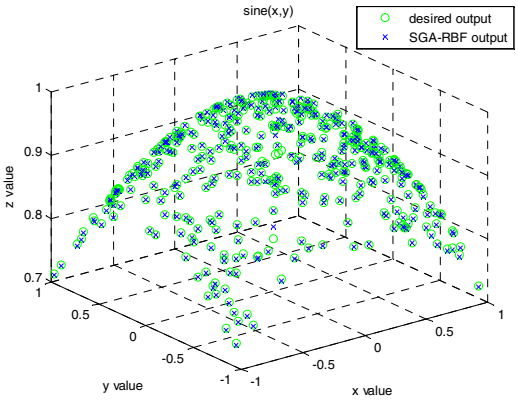

Fig. 9. The testing results

C. *Nonlinear system modeling*

One of the nonlinear dynamic systems is given by (25), which has been used in several literatures, notably [35], [36], to demonstrate the effectiveness of RBF algorithms

$$y(t+1)=\frac{y(t)\,y(t-1)\left[y(t)+2.5\right]}{1+y^2(t)+y^2(t-1)}+u(t), \qquad (25)$$

where $t \in [1,500]$ $y(0)=0$, $y(1)=0$, $u(t)=sin(2\pi t/25)$.

The model is defined as follows:

$$\hat{y}(t+1)=f\left(y(t),y(t-1),u(t)\right). \qquad (26)$$

There are three inputs (y(t), y(t-1), u(t)) and one output $\hat{y}$ (t+1) in the SGA-RBF neural network. In the training phase, A set of 400 points were chosen between t=1 and 400 according to (26) as training data. Another 100 input-target points in the interval [401, 500] were used as the testing data. The initial number of hidden neurons was 2. The pre-set training error value was 0.01. The centers, the radius and weights were generated randomly in a small range. The experiment results are presented in Figs. 10-11 and Table 2.

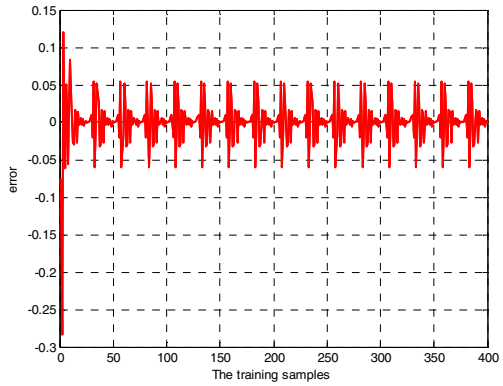

Fig. 10. The training error

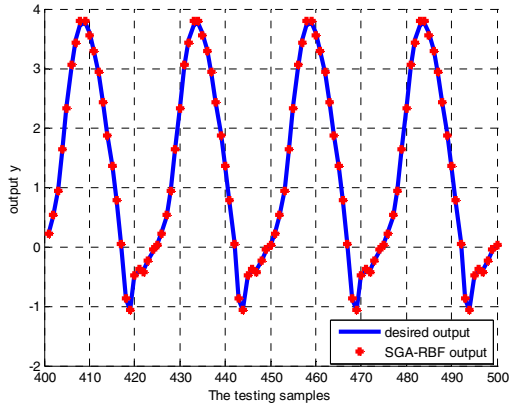

Fig. 11. The testing output

Table.3 The performance comparison of different algorithms

Algorithms	*Hidden neurons*	*Running time(s)*	*RMSE*	
			training	*testing*
SGA-RBF	**23**	**0.92**	**0.0175**	**0.0166**
RBF	54	104.38	0.0243	0.0194
OLS[37]	65	1.91	0.0288	0.4452
RBFAFS[37]	35	*	0.1384	*
Farag's model[37]	75	*	0.1930	0.2010

*The results are no listed in original papers.

The number of RBF algorithm was dynamically chosen from the consecutive integer 1 to 80. The RBF neural network with 54 hidden neurons has the smallest training and testing RMSE values.

Fig.10 shows the error values in the training process. The error values are in a small range periodically except the first few training data. The testing outputs of the SGA-RBF neural network are shown to make a comparison with the desired output values in Fig.11. The testing output values of the SGA-RBF neural network are nearly equal to the desired outputs. The performance comparison with other algorithms is presented in Table 3.

Table 3 shows that the proposed SGA-RBF neural network not only has the smallest number of hidden neurons in the listed algorithms but has a smaller training and testing RMSE values. The smaller number of hidden neurons means that the proposed SGA obtains a more compact network. The better

RMSE value in the testing phase suggests that the proposed SGA-RBF has better generalization ability. Moreover, the training time is much shorter than the dynamic RBF neural network and OLS algorithm, which also demonstrates the conclusion obtained in Section 4 that the time cost used in SGA algorithm is less than OLS method. It indirectly again implies that the proposed SGA is efficient to design RBF neural networks, which works in a plausible biological mode.

V. Conclusion

This paper proposed a SGA for self-organizing RBF neural network. The proposed SGA-RBF neural network derives from the cerebral cortex aims to obtain a compact structure and precise operation. The simulation results show that the proposed SGA-RBF neural network is suitable to approximate the nonlinear functions and model the nonlinear systems. From the experimental results, we can safely draw these conclusions:

1) The proposed SGA-RBF neural network can realize the dynamic adjustment with a result of self-organizing structure.

2) The growing mechanism, determined both by the *ss* of the hidden neurons and the errors of the training process, can improve the accuracy of the SGA-RBF neural network.

3) In the training phase, parameters of the new inserted neurons were redefined. The experiment results prove that oscillation of the errors weakened by the compensation.

4) Three experiments revealed that the SGA-RBF neural network has a satisfactory generalization power.

In the future work, a pruning algorithm will be designed for the SGA-RBF neural network. In addition, some effective learning algorithms will be considered for training the parameters.

Acknowledgment

This work was supported by the National Science Foundation of China under Grant 61203099; National Science Foundation of China under Grant 61034008; Ph.D. Programs Foundation of Ministry of Education of China under Grant 20121103120020; Beijing Municipal Natural Science Foundation under Grant 4122006; Beijing Nova Program under Grant Z131104000413007; Hong Kong Scholor Program under Grant XJ2013018.

References

[1] M. Bortman, M. Aladjem, "A growing and pruning method for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 1039–1045, June 2009.

[2] H. G. Han, J. F. Qiao, "Adaptive computation algorithm for RBF neural network," *IEEE Transactions on Neural Networks*, vol. 23, no. 2, pp. 342–347, February 2012.

[3] J. Moody, C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, no. 2, pp. 281–294, March 2008.

[4] S. Ferrari, F. Bellocchio, V. Piuri, "A hierarchical RBF online learning algorithm for real-time 3-D scanner," *IEEE Transactions on Neural Networks*, vol. 21, no. 2, pp. 275–282, February 2010.

[5] W.K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[6] T. T. Xie, Y. Hao, H. Joel, "Fast and efficient second-order method for training radial basis function networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 4, pp. 609–619, April 2012.

[7] S. Chen, C. F. N. Cowan, P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 302–309, March 1991.

[8] S. Venkatesh, S. Gopal," Orthogonal least square center selection technique- A robust scheme for multiple source partial discharge pattern recognition using radial basis probabilistic neural network," Expert Systems with Applications, vol. 38, no. 7, pp. 8978–8989, July 2011.

[9] S. Chen, J. Wigger, "Fast orthogonal least squares algorithm for efficient subset model selection," *IEEE Transactions on Signal Processing*, vol. 43, no. 7, pp. 1713–1715, July 1995.

[10] B. Walczak, D. L. Massart, "The radial basis functions-partial least squares approach as a flexible non-linear regression technique," *Analytica Chimica Acta*, vol. 331, no. 3, pp. 177–185, September 1996.

[11] J. B. Gomm, D. L. Yu, "Selecting radial basis function network centers with recursive orthogonal least squares training," *IEEE Transactions on Neural Networks*, vol. 2, no. 11, pp. 306–314, March 2000.

[12] D. Shi, D. S. Yeung, J. Gao, "Sensitivity analysis applied to the construction of radial basis function networks," *Neural Networks*, vol. 18, no. 7, pp. 951–957, September 2005.

[13] B. Sudret, "Global sensitivity analysis using polynomial chaos expansions," *Reliability Engineering and System Safety*, vol. 93, no. 7, pp. 964–979, July 2008.

[14] A. Saltelli, S. Tarantola, K. Chan, "A quantitative, model independent method for global sensitivity analysis of model output," *Technometrics*, vol. 41, no. 1, pp. 39–56, March 1999.

[15] J. F. Qiao, H. G. Han, "A repair algorithm for radial basis function neural network and its application to chemical oxygen demand modeling," *International Journal of Neural Systems*, vol. 20, no. 1, pp. 63–74, January 2010.

[16] J. Platt, "A resource allocating network for function interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, March 1991.

[17] V. Kadirkamanathan, M. Niranjan, "A function estimation approach to sequential learning with neural networks," *Neural Computation*, vol. 5, no. 6, pp. 954–975, November 1993.

[18] Y. Lu, N. Sundararajan, P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function neural networks," *Neural Computation*, vol. 9, no. 2, pp. 461–478, February 1997.

[19] L. Yingwei, N. Sundararajan, P. Saratchandran, "Identification of time-varying nonlinear systems using minimal radial basis function neural networks," *IEEE Proceedings Part D-Control Theory and Applications*, vol. 144, no. 1, pp. 1–7, March 1997.

[20] G. B. Huang, P. Saratchandran, N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE Transactions on Neural Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 34, no. 6, pp. 2284–2292, December 2004.

[21] G. B. Huang, P. Saratchandran, N. Sundararajan, "A generalized growing and pruning RBF(GGAP-RBF) neural network for function approximation," *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 57–67, January 2005.

[22] A. Alex, H. Sarimveis, B. George, "A new algorithm for online structure and parameter adaptation of RBF networks," *Neural Networks,* vol. 16, no. 7, pp. 1003–1017, September 2003.

[23] K. P. Ferentions, "Biological engineering applications of feedforward neural networks designed and parameterized by genetic algorithms," *Neural Networks*, vol. 18, no. 7, pp. 934–950, September 2005.

[24] J. J. Yao, J. Yang, L. M. Wang, "A HAMPSO-RBF algorithm applied to target localization," *AASRI Conf. Computational Intelligence and Bioinformatics*, vol. 1, pp. 183–188, June 2012.

[25] A. Alexandridis, E. Chondrodima, H. Sarimveis, "Radial basis function network training using a nonsymmetric partition of the input space and particle swarm optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 2, pp. 219–230, February 2013.

[26] Y. Xu, X. Q. Zeng, L. X. Han, J. Yang, "A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks," *Neural Networks*, vol. 43, no. 2, pp. 99–113, February 2013.

[27] C. Glackin, L. Maguire, L. McDaid, J. Wade, "Synchrony: A spiking-based mechanism for processing sensory stimuli," *Neural Networks*, vol. 32, no. 2, pp. 26–34, August 2012.
[28] S. Alexander, I. Kastalskiy, V. Kazantsev, "Pattern retrieval in a three-layer oscillatory network with a context dependent synaptic connectivity," *Neural Networks,* vol. 33, no. 4, pp. 67–75, September 2012.
[29] M. Islam, A. Sattear, F. Amin, X. Yao, K. Murase, "A new constructive algorithm for architectureal and functional adaptation of artificial neural networks," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics,* vol. 39, no. 6, pp. 1590–1605, December 2009.
[30] Y. W. Lu, N. Sundararajan, P. Saratchandran, "Performance evaluation of a sequential minimal radial basis function(RBF) neural network learning algorithm," *IEEE Transactions on Neural Networks*, vol. 9, no. 2, pp. 308–318, March 1998.
[31] H. G. Han, J. F. Qiao, "Prediction of activated sludge bulking based on a self-organizing RBF neural network," *Journal of Process Control,* vol. 22, no. 6, pp. 1103–1112, July 2012.
[32] H. G. Han, J. F. Qiao, "Identification and modeling of nonlinear dynamic systems using a novel self-organizing RBF-based approach," *Automatica*, vol. 48, no. 8, pp. 1729–1734, June 2012.
[33] D. J. C. Mackay, "Bayesian Interpolation," *Neural Computation,* vol. 4, no. 3, pp. 415–447, May 1992.
[34] P. Zhou, D. H. Li, H. Wu, F. Cheng, "The automatic model selection and variable kernel width for RBF neural networks," *Neurocomputing*, vol. 74, no. 17, pp. 3628–3637, October 2011.
[35] R. Enrique, M. S. Josep, "Performing feature selection with multilayer perceptrons," *IEEE Transactions on Neural Networks*, vol. 19, no. 3, pp. 431–441, March 2008.
[36] G. B. Huang, P. Saratchandran, N. Sundararajan, "A generalized growing and pruning RBF(GGAP-RBF) neural network for function approximation," *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 57–67, January 2005.
[37] G. Leng, T. M. McGinnity, G. Prasad, "Design for self-organizing fuzzy neural networks based on genetic algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 14, no. 6, pp. 755–766, December 2006.