An Accelerated PSO Based Self-organizing RBF Neural Network for Nonlinear System Identification and Modeling

Zohaib Ahmad(✉), Cuili Yang, and Junfei Qiao

Faculty of Information Technology, Beijing Key Laboratory of Computational Intelligence and Intelligent System, Beijing University of Technology, Beijing 100124, China
ahmedzohaib03@gmail.com,
{clyang5,junfeiq}@bjut.edu.cn

Abstract. In this paper, an accelerated particle swarm optimization (APSO) based radial basis function neural network (RBFNN) is designed for nonlinear system modeling. In APSO-RBFNN, the center, width of hidden neurons, weights of output layer and network size are optimized by using the APSO method. Two nonlinear system modeling experiments are used to illustrate the effectiveness of the proposed method. The simulation results show that the proposed method has obtained good performance in terms of network size and estimation accuracy.

Keywords: Accelerated particle swarm optimization · Radial basis function · Nonlinear system modeling

1 Introduction

Due to the simple topological structure and rapid training speed, the radial basis function neural networks (RBFNNs) have been widely used in many applications, such as the nonlinear system estimation [1], information processing [2], pattern recognition [3], and so on. For a RBF neural network, its performance is closely related with network architecture, thus how to determine the optimal network parameters and network size is very important.

If the network parameters are selected in a random manner, the network satisfactory performance cannot be achieved. To solve this problem, many different learning algorithms are designed to adjust network parameters. As the most commonly used method, the back propagation (BP) has been used to adjust RBFNN parameters [3]. However, the BP method always results in degraded global search experience and is easily trapped into local optimal [3]. Compared with BP method, the recursive least squares (RLS) methods can result in a more optimum convergence rate and improved network performance [4]. However, the application of RLS always needs the complicated mathematical expression and high computing resources.

The network size is also closely related with network performance of RBFNN. Hence, it is important to optimize the size of RBFNN to improve network performance. For example, the adaptive growing and pruning algorithm [5] is applied to construct the

X. B. Zhai et al. (Eds.): MLICOM 2019, LNICST 294, pp. 769–777, 2019.
https://doi.org/10.1007/978-3-030-32388-2_63

recurrent neural network, and the neurons are added or removed to improve network performance. Then, the forward and backward selection methods are used for the self-organizing RBFNN [6], in which the network size and parameters are optimized simultaneously.

Recently, the evolutionary algorithms (EA) are widely used to improve network performance. There have existed many EA algorithms, such as the differential evolution (DE), simulated annealing (SA), artificial immune algorithm (IA), particle swarm optimization (PSO), ant colony algorithm (AC), and genetic algorithm (GA). Among these algorithms, the PSO offers quick convergence rate and good global searching ability. However, the traditional PSO always suffer from weak solution diversity and premature convergence for highly nonlinear optimization problem. To solve this problem, the accelerated PSO (APSO) [7] is proposed which has shown superiority in terms of algorithm convergence speed and solution accuracy.

Based on above discussion, in this paper, the APSO is proposed to optimize network parameters and network size of RBFNN. Then, the developed network is compared with some existing methods. The simulation results show that the proposed method is efficient in terms of estimation accuracy and network compactness.

The rest of paper has been presented as follows: In Sect. 2, the RBFNN is discussed. Section 3 overviews the PSO and APSO. In Sect. 4, the simulation results have been depicted. Finally, the conclusion is presented in Sect. 5.

2 RBF Neural Network Model

The RBFNN contains three layers, including the input layer, hidden layer and output layer. The basic illustration of a RBFNN is presented in Fig. 1.

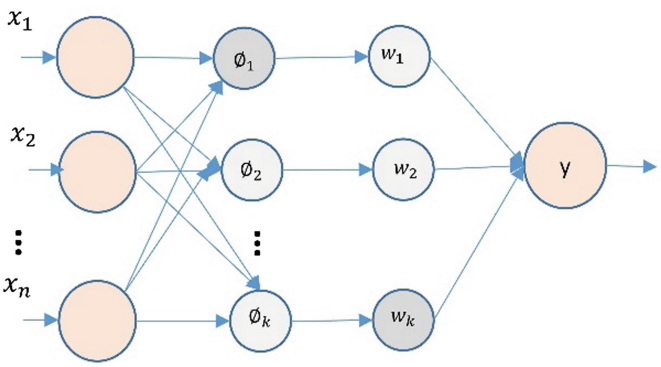

Fig. 1. The RBFNN network illustration

The output of RBFNN, $\boldsymbol{y}(t)$, is specified by the given equation:

$$\boldsymbol{y}(t) = \sum_{l=1}^{K} w_l(t)\phi_l(t); l = 1, 2, 3, \ldots, K \tag{1}$$

Where $w_l(t)$ is the weight of output at the time step t, l is the hidden neuron with $l = 1, 2, 3, \ldots, K$, $\phi_l(t)$ is the lth hidden neuron output as below:

$$\phi_l(t) = e^{\|\boldsymbol{x}(t) - \mu_l(t)\| / \sigma_l^2(t)} \tag{2}$$

where $\boldsymbol{x}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^{\mathrm{T}}$ are inputs of the RBFNN, n is the nodes number of input layer; $\boldsymbol{\mu}_l(t)$ indicates the center vector of the lth hidden neuron; $\|\boldsymbol{x}(t) - \boldsymbol{\mu}_l(t)\|$ is the Euclidean distance among $\boldsymbol{x}(t)$ and $\boldsymbol{\mu}_l(t)$; $\sigma_l(t)$ is the radius or width of the lth hidden neuron. The network parameters, including width, center of hidden neurons, output weights and network size are adjusted in the training phase by using the training samples.

3 The Proposed APSO-RBFNN

3.1 Overview of PSO

In the traditional PSO [7], the swam is defined as a set $\mathrm{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_N\}^{\mathrm{T}}$ of N particles. Each particle $\boldsymbol{s}_i(k)$ at the iteration k is assigned a position vector, which is represented as below,

$$\boldsymbol{s}_i(k) = [s_{i,1}(k), s_{i,2}(k), \ldots, s_{i,D}(k)] \tag{3}$$

where $i = 1, 2, \ldots, N$, and D stands for the search space dimension. Moreover, each particle has a velocity $\boldsymbol{v}_i(k)$ which is denoted by:

$$\boldsymbol{v}_i(k) = [v_{i,1}(k), v_{i,2}(k), v_{i,3}(k), \ldots, v_{i,D}(k)] \tag{4}$$

In order to allow each particle visit all the searching space, the velocity of each particle is updated according to its gained memory. Now, let us define $\boldsymbol{p}_i(k)$ as the best previous position of the ith particle, which is denoted as $\boldsymbol{p}_i(k) = [p_{i,1}(k), p_{i,2}(k), \ldots, p_{i,D}(k)]$. Then, the best position found by the whole swarm is recorded as $\boldsymbol{g}_{best}(t) = [g_{best,1}(k), g_{best,2}(k), \ldots, g_{best,D}(k)]$.

At each iteration k, the velocity of each particle is updated as:

$$\boldsymbol{v}_i(k+1) = \omega v_i(k) + c_1 r_1 (\boldsymbol{p}_{best}(k) - \boldsymbol{s}_i(k)) + c_2 r_2 (\boldsymbol{g}_{best}(k) - \boldsymbol{s}_i(k)) \tag{5}$$

where ω is the weight of inertia; r_1 and r_2 are random values derived from the interval [0,1], which are used to maintain the algorithm diversity; c_1 and c_2 are acceleration constants, which are used to push the swarm towards the local and global best position. Finally, the position of each particle is updated as below:

$$\boldsymbol{s}_i(k+1) = \boldsymbol{v}_i(k+1) + \boldsymbol{s}_i(k) \tag{6}$$

3.2 Overview of APSO

Based on the traditional PSO, the accelerated particle swarm optimization (APSO) is proposed [7]. In traditional PSO, the particle best position $\boldsymbol{p}_i$ and the global best position $\boldsymbol{g}_{best}$ are used to update the velocity as shown in Eq. (6). While in the APSO, only the global best position $\boldsymbol{g}_{best}$ are used. Consequently, the particle's velocity in APSO is updated as below:

$$\boldsymbol{v}_i(k+1) = \boldsymbol{v}_i(k) + \alpha \boldsymbol{q} + \beta(\boldsymbol{g}_{best} - \boldsymbol{s}_i(k)) \tag{7}$$

where $\boldsymbol{q}$ is a random number chosen from the range [0,1]. Generally, the parameter α is chosen from the range $[0.1D, 0.5D]$, and β can be chosen from the range [0.1, 0.7]. Moreover, the position of each particle is simply calculated as below:

$$\boldsymbol{s}_i(k+1) = (1-\beta)\boldsymbol{s}_i(k) + \beta \boldsymbol{g}_{best} + \alpha \boldsymbol{q} \tag{8}$$

As illustrated in [7], compared with traditional PSO, the APSO has shown good performance in terms of algorithm convergence and diversity.

3.3 Design Process of the Proposed APSO-RBFNN

In this paper, the APSO is used to optimize network parameters of RBFNN, the proposed method is denoted as APSO-RBFNN. In APSO-RBFNN, the network parameters, including width $\boldsymbol{\mu}_l$ and center $\boldsymbol{\sigma}_l$ of hidden neurons, output weights w_l and network size are optimized by the APSO. The operation procedures of APSO-RBFNN are given as follows:

Step 1: Set the original iteration as $k = 0$. Randomly initialize the position $\boldsymbol{s}_i$ and velocity v_i of each particle i.
Step 2: Evaluate the fitness function for each particle i. Among all the particles, find $\boldsymbol{g}_{best}$ with the best fitness value.
Step 3: Calculate the positions and velocities of all particles according to Eqs. (7) and (8).
Step 4: Start the next generation $k = k+1$ and go to Step 2. This process is repeated until the maximum iteration is reached.

4 Simulation Results

In the simulation part, the root-mean square error (RMSE) is used to test the network performance, it is also treated as the objective function of APSO. The mathematical function of RMSE is given as follows:

$$E_i(k) = \sqrt{\frac{1}{L}\sum\nolimits_{t=1}^{L} (y_d(t) - \hat{y}(t)^2)} \tag{9}$$

where, y_d is the desired output, L represents data samples length and $\hat{y}$ is the network's estimated output. The smaller the RMSE value, the better the neural network estimation performance. To demonstrate efficiency of the designed APSO- RBFNN, two non-linear system modeling experiments are illustrated.

4.1 The Mackey-Glass Time-Series Estimation

As a classical benchmark task for non-linear system modeling, the Mackey-Glass time-series are generated as below [5] (Fig. 2):

$$x(t+1) = (1-a)x(t) + \frac{bx(t-\tau)}{1+x^{10}(t-\tau)} \tag{10}$$

where the parameters are set as a = 0.1, b = 0.2, τ = 17, and the initial condition is $x(0)$ = 0. In this experiment, 1000 samples are set as the training set, the first 500 data are discarded to reduce the influence of initial condition. The samples with t = 1001 to 1500 are set as the testing set. The simulation results of the proposed APSO-RBFNN are shown in Figs. 3, 4 and 5. In Fig. 3, the comparisons between the estimated outputs and target are shown. In Fig. 4, the evolution process of the hidden neurons is illustrated. While in Fig. 5, the testing RMSE curve is displayed. From Figs. 3, 4 and 5, it can be concluded that the proposed APSO-RBFNN has obtained good estimation result.

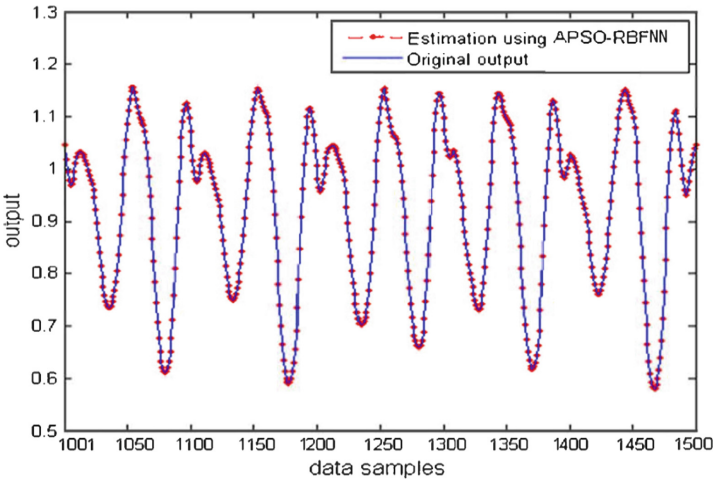

Fig. 2. The testing results of APSO-RBFNN for Mackey-Glass time series

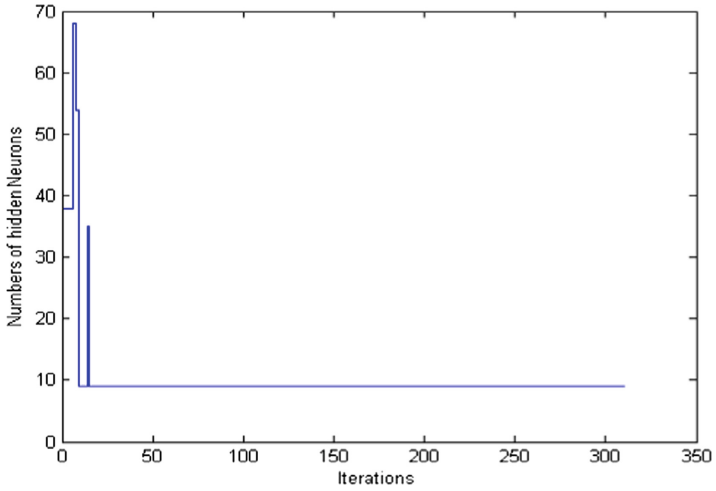

Fig. 3. The evolution of neurons versus iterations for APSO-RBFNN

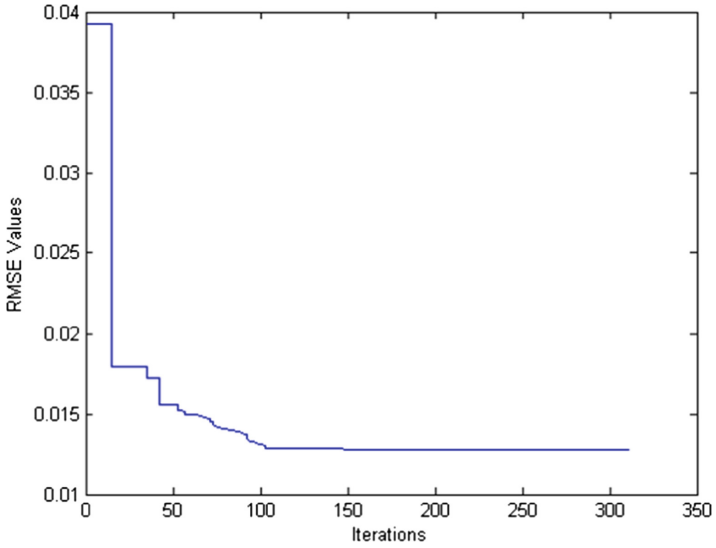

Fig. 4. The RMSE value evolving process versus iterations for APSO-RBFNN

In order to test the effectiveness of the proposed APSO-RBFNN, its performance is compared with other two methods, i.e., the PSO-RBFNN in which the network parameters are optimized by PSO, the DE-RBFNN whose network parameters are optimized by differential algorithm (DE). As shown in Table 1, the APSO-RBFNN owns the optimal compact network (10 hidden neurons), the lowest training RMSE of 0.0107 and the lowest testing RMSE of 0.0127.

Table 1. Performance of various methods for Mackey-Glass time-series estimation

Algorithm	RMSE (training)	RMSE (testing)	No. of hidden neurons
APSO-RBFNN	0.0107	0.0127	10
PSO-RBFNN	0.0256	0.0208	12
DE-RBFNN	0.0192	0.0189	13

4.2 Lorenz Time-Series Prediction

The Lorenz time-series prediction is another benchmark to evaluate the nonlinear system modeling performance for neural networks. Its differential equations are presented as below:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= rx - y - xz, \\ \frac{dz}{dt} &= xy - bz \end{aligned} \tag{11}$$

where $\sigma = 10, b = 8/3, r = 28$, the initial conditions for each variable are set as $x(0) = y(0) = z(0) = 0.1$. A total of 5000 data samples are generated, the data samples with t = 1 to 3000 are set as the training set and the remaining part is set as the testing set. In the training set, the first 1000 samples are discarded.

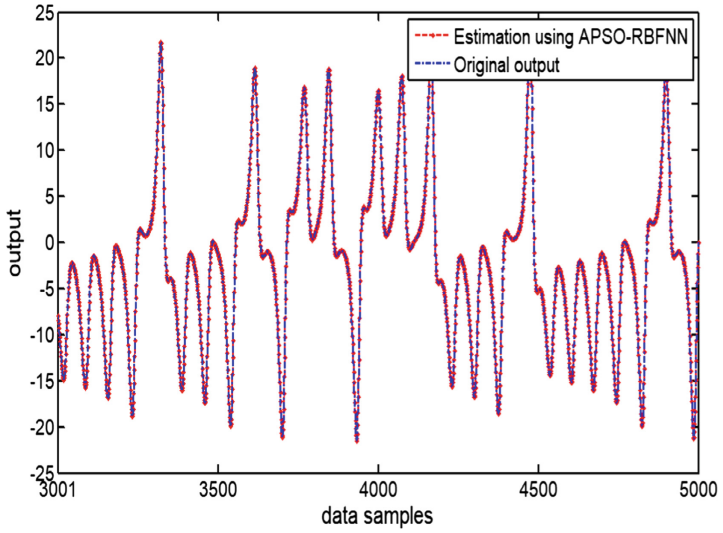

Fig. 5. The testing results of APSO-RBFNN for Lorenz time series prediction

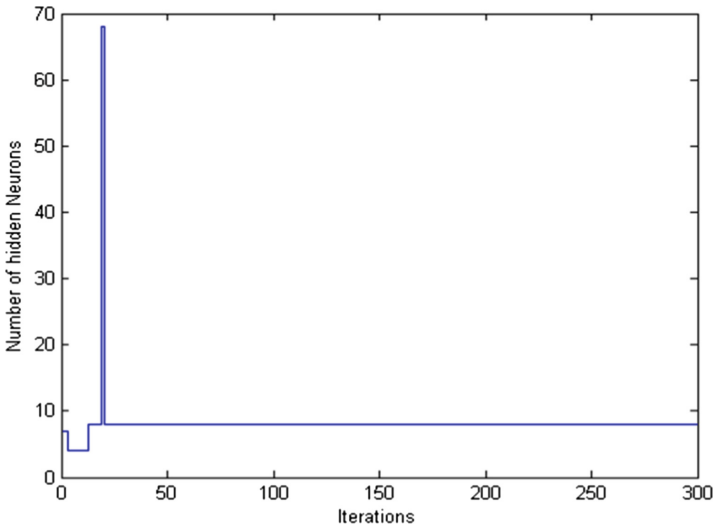

Fig. 6. The network size evolution process versus iterations for APSO-RBFNN

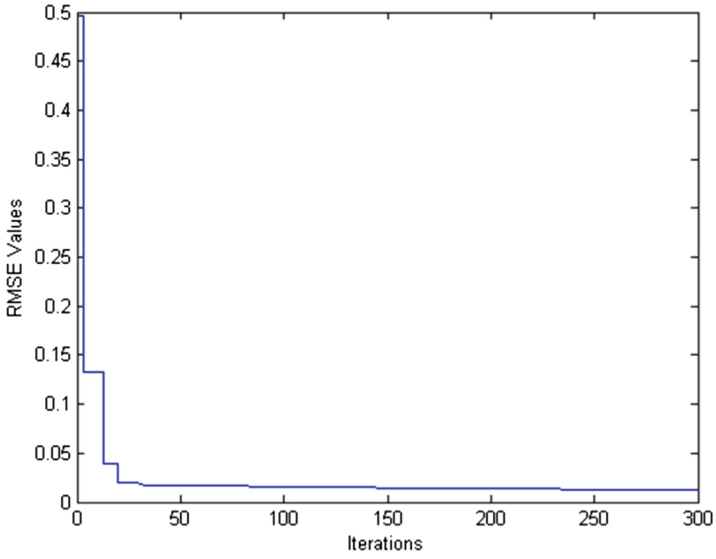

Fig. 7. The RMSE values versus algorithm iterations for APSO-RBFNN

The estimation results on the testing set of APSO-RBFNN are shown in Fig. 5, the evolving process of the network size is illustrated in Fig. 6, and the RMSE values versus algorithm iterations are given in Fig. 7. Furthermore, the performance of the proposed APSO-RBFNN is compared with PSO-RBFNN and DE-RBFNN. The

comparison details are given in Table 2, it is easily found that the designed APSO-RBFNN has better training and testing RMSE as well as network size than the other two methods.

Table 2. Performance of various methods for Lorenz time series estimation

Algorithm	RMSE (training)	RMSE (testing)	No. of hidden neurons
APSO-RBFNN	0.0131	0.0126	8
PSO-RBFNN	0.2132	0.2677	9
DE-RBFNN	0.1900	0.2017	10

5 Conclusion

To solve the nonlinear system modeling problem, an accelerated particle swarm optimization (APSO) based radial basis function neural network (RBFNN) is proposed, which is denoted as APSO-RBFNN. In APSO-RBFNN, the center, width of hidden neurons, output weights and network size are optimized by using the APSO method. As shown in the simulation results, the proposed APSO-RBFNN outperforms other existing method in terms of network compactness and estimation accuracy.

Acknowledgement. This work was supported by the National Natural Science Foundation of China under Grants 61603012, 61533002 and 61890930-5, the Beijing Municipal Education Commission Foundation under Grant KM201710005025, the Major Science and Technology Program for Water Pollution Control and Treatment of China (2018ZX07111005), the National Key Research and Development Project under Grants 2018YFC1900800-5.

References

1. Jian, Y.E., Lindong, G.E., et al.: An application of improved RBF neural network in modulation recognition. Acta Automatica Sinica **33**(6), 652–654 (2007)
2. Chen, S., Wolfgang, A., Harris, C.J., et al.: Symmetric RBF classifier for nonlinear detection in multiple-antenna-aided systems. IEEE Trans. Neural Netw. **19**(5), 737–745 (2008)
3. Xu, Z., Zhang, R., Jing, W.: When does online BP training converge? IEEE Trans. Neural Netw. **20**(10), 1529–1539 (2009)
4. Chen, B., Zhao, S., Zhu, P., Príncipe, J.C.: Quantized kernel recursive least squares algorithm. IEEE Trans. Neural Netw. Learn. Syst. **24**(9), 1484–1491 (2013)
5. Han, H.G., Zhang, S., Qiao, J.F.: An adaptive growing and pruning algorithm for designing recurrent neural network. Neurocomputing **242**, 51–62 (2017)
6. Qiao, J.F., Han, H.G.: Identification and modeling of nonlinear dynamical systems using a novel self-organizing RBF-based approach. Automatica **48**(8), 1729–1734 (2012)
7. Guedria, N.B.: Improved accelerated PSO algorithm for mechanical engineering optimization problems. Appl. Soft Comput. **40**(40), 455–467 (2016)