Neural Comput & Applic (2010) 19:255–261
DOI 10.1007/s00521-009-0274-y

ORIGINAL ARTICLE

An improved approximation approach incorporating particle swarm optimization and a priori information into neural networks

Fei Han · Qing-Hua Ling · De-Shuang Huang

Received: 5 August 2008 / Accepted: 15 April 2009 / Published online: 30 April 2009

Abstract In this paper, an improved approach incorporating adaptive particle swarm optimization (APSO) and a priori information into feedforward neural networks for function approximation problem is proposed. It is well known that gradient-based learning algorithms such as backpropagation algorithm have good ability of local search, whereas PSO has good ability of global search. Therefore, in the improved approach, the APSO algorithm encoding the first-order derivative information of the approximated function is used to train network to near global minima. Then, with the connection weights produced by APSO, the network is trained with a modified gradient-based algorithm with magnified gradient function. The modified gradient-based algorithm can reduce input-to-output mapping sensitivity and lessen the chance of being trapped into local minima. By combining APSO with local search algorithm and considering a priori information, the improved approach has better approximation accuracy and convergence rate. Finally, simulation results are given to verify the efficiency and effectiveness of the proposed approach.

F. Han (✉)
School of Computer Science and Telecommunication Engineering, Jiangsu University, 212013 Zhenjiang, Jiangsu, China
e-mail: hanfei1976@163.com

Q.-H. Ling
School of Computer Science and Engineering, Jiangsu University of Science and Technology, 212003 Zhenjiang, Jiangsu, China

D.-S. Huang
Intelligent Computing Lab, Hefei Institute of Intelligent Machines, Chinese Academy of Sciences, P.O.Box 1130, 230031 Hefei, Anhui, China

Keywords Function approximation · Particle swarm optimization · A priori information · Approximation accuracy · Convergence rate

1 Introduction

The popular method for training a feedforward neural network (FNN) is the backpropagation (BP) algorithm [1]. However, the traditional BP algorithm has the following major drawbacks. First, it is apt to be trapped in local minima. Second, it have not considered the network structure features as well as the involved problem properties, thus its generalization capabilities are limited. Finally, since BP algorithms are the gradient-based type learning algorithms, they converge very slowly [2–7].

In the literatures [8, 9], two algorithms referred as Hybrid-I and Hybrid-II methods, respectively were proposed. The Hybrid-I algorithm incorporates the first-order derivatives of the neural activation at hidden layers into the sum-of-square error cost function to reduce the input-to-output mapping sensitivity. The Hybrid-II algorithm incorporates the second-order derivatives of the neural activations at hidden layers and output layer into the sum-of-square error cost function to penalize the high-frequency components in training data. In the literature [10], a modified hybrid algorithm (MHLA) is proposed according to Hybrid-I and Hybrid-II algorithms to improve the generalization performance. All the above learning algorithms are purely local search algorithms and apt to converge to local minima.

Obviously, gradient-based learning algorithm has good capability of local search. On the other hand, particle swarm optimization (PSO) algorithm has good capability of global search [11–15]. Therefore, global search combining with local search in a learning algorithm can improve the convergence performance of the algorithm.

In the recent years, PSO has been used increasingly as an effective technique for searching global minima [13–16]. When compared to genetic algorithm, the PSO algorithm has no complicated evolutionary operators and adjusts less parameter in the course of training [17–19].

Hence, in the literature [20], a double search approach referred as APSOAEFDI–MHLA for function approximation was proposed to obtain better approximation performance. First, the APSOAEFDI which combined APSO with the first-order derivative information of the approximated function was used to search globally. Then MHLA was used to search locally within the global search. In this paper, an improved approach similar to APSOAEFDI–MHLA for function approximation based on adaptive particle swarm optimization (APSO) and a priori information is proposed. In order to overcome the drawbacks from gradient-based algorithm for FNN, the FNN is trained by the APSOAEFDI first to near the global minima, and then the network is trained again by a modified gradient-based algorithm with magnified gradient function [21]. Due to combining APSO with local search algorithm and considering the a priori information, the improved approach has better approximation accuracy and convergence rate. Finally, simulation results are given to verify the efficiency and effectiveness of the proposed learning approach.

2 Particle swarm optimization

The PSO is an evolutionary computation technique developed by Eberhart and Kennedy in 1995 [11, 12], inspired by social behavior of bird flocking. PSO is a kind of algorithm to search for the best solution by simulating the movement of flocking of birds. The algorithm works by initializing a flock of birds randomly over the searching space, where each bird is called as a “particle”. These “particles” fly with a certain velocity and find the global best position after some iteration. At each iteration, each particle can adjust its velocity vector, based on its momentum and the influence of its best position (P_b) as well as the best position of its neighbors (P_g), and then compute a new position that the “particle” is to fly to. Supposing the dimension of searching space is D, the total number of particles is n, the position of the ith particle can be expressed as vector $X_i = (x_{i1}, x_{i2},\ldots, x_{iD})$; the best position of the ith particle searching until now is denoted as $P_{ib} = (p_{i1}, p_{i2},\ldots, p_{iD})$, and the best position of the total particle swarm searching until now is denoted as vector $P_g = (p_{g1}, p_{g2},\ldots, p_{gD})$; the velocity of the ith particle is represented as vector $V_i = (v_{i1}, v_{i2},\ldots, v_{iD})$. Then the original PSO algorithm (PSOA) [11, 12] is described as:

$$v_{id}(t+1) = v_{id}(t) + c_1 \times \text{rand}() \times [p_{id}(t) - x_{id}(t)] + c_2 \times \text{rand}() \times [p_{gd}(t) - x_{id}(t)] \tag{1}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad 1 \le i \le n \quad 1 \le d \le D \tag{2}$$

where c_1, c_2 are the acceleration constants with positive values; rand() is a random number between 0 and 1. In addition to the c_1, and c_2 parameters, the implementation of the original algorithm also requires to place a limit on the velocity ($v_{\max}$). After adjusting the parameters w and $v_{\max}$, the PSO can achieve the best search ability.

The adaptive particle swarm optimization algorithm (APSOA) is based on the original PSO algorithm, proposed by Shi and Eberhart in 1998 [22, 23]. The APSO can be described as follows:

$$v_{id}(t+1) = w \times v_{id}(t) + c_1 \times \text{rand}() \times [p_{id}(t) - x_{id}(t)] + c_2 \times \text{rand}() \times [p_{gd}(t) - x_{id}(t)] \tag{3}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad 1 \le i \le n \quad 1 \le d \le D \tag{4}$$

where w is a new inertial weight. The parameter w can reduce gradually as the generation increases. The APSO algorithm is more effective, because the searching space reduces step by step, not linearly.

3 Modified Hybrid-I algorithm with magnified gradient function

Above all, the following mathematical notations are made. x_k and y_i denote the kth element of the input vector and the ith element of the output vector, respectively; $w_{j_l j_{l-1}}$ denotes the synaptic weight from the j_{l-1} th hidden neuron at the $(l-1)$th hidden layer to the j_l th hidden neuron at the lth hidden layer; $w_{ij_{L-1}}$ denotes the synaptic weight from the j_{L-1} th hidden neuron at the $(L-1)$th hidden layer to the ith neuron at the output layer; $w_{j_1 k}$ denotes the synaptic weight from the kth element of the input vector to the j_l th hidden neuron at the first hidden layer; $f_l'(\cdot)$ is the derivative of the activation function $f_l(\cdot)$ at the lth hidden layer; $h_{j_l} = f_l(\hat{h}_{j_l})$ is the activation function of the j_l th element at the lth hidden layer with $\hat{h}_{j_l} = \sum_{j_{l-1}} w_{j_l j_{l-1}} h_{j_{l-1}}$. The t_i and y_i denote the target and actual output values of the ith neuron at output layer, respectively; N_l denotes the number of the neurons at the lth hidden layer; N_L denotes the number of the neurons at the output layer.

In order to reduce the input-to-output mapping sensitivity, a cost function in Hybrid-I [8, 9] has been proposed as follows:

$$E = \frac{1}{N}\sum_{S=1}^{N} E^S = \frac{1}{N}\sum_{S=1}^{N}\left(\frac{1}{2N_L}\sum_{i=1}^{N_L}\left(t_i^S - y_i^S\right)^2 + \sum_{l=1}^{L-1}\gamma_l E_h^{lS}\right) \tag{5}$$

where $E_h^{lS} = \frac{1}{N_l}\sum_{j_l}^{N_l} f'\left(\hat{h}_{j_l}^S\right)$ denotes the additional hidden layer penalty term at lth layer; The gain γ_l represents the relative significance of the hidden layer cost over the output error.

The network is trained by a steepest-descent error minimization algorithm as usual, and the synaptic weight update for the Sth stored pattern becomes [8, 9]

$$\Delta w_{j_l j_{l-1}}^S = -\eta_l \frac{\partial E^S}{\partial w_{j_l j_{l-1}}^S} = \eta_l\, \delta_{j_l}^S\, h_{j_{l-1}}^S \qquad l = 1, 2, \ldots, L \tag{6}$$

where $\delta_{j_l}^S$ denotes the negative derivative of E^S to $\hat{h}_{j_l}^S$ at the lth layer.Hence, the negative derivative of E^S to $\hat{h}_{j_l}^S (l = 1, \ldots, L-1, L)$ at the hidden layer for the Sth stored pattern, i.e., $\delta_{j_l}^S$, can be computed by back-propagation style as follows [8, 9]:

$$\delta_{j_l}^S = -\frac{\partial E^S}{\partial \hat{h}_{j_l}^S} = \sum_{j_{l+1}=1}^{N_{l+1}} \delta_{j_{l+1}}^S\, w_{j_{l+1} j_l}^S f'\left(\hat{h}_{j_l}^S\right) - \frac{\gamma_l}{N_l} f''\left(\hat{h}_{j_l}^S\right) \tag{7}$$
$$l = 1, \ldots, L-1$$

$$\delta_{j_L}^S = -\frac{\partial E^S}{\partial \hat{h}_{j_L}^S} = \frac{1}{N_L} f'\left(\hat{h}_{j_L}^S\right)\left(t_i^S - y_i^S\right) \tag{8}$$

In this paper, we adopt the activation function for all hidden neurons at all layers, i.e., tangent sigmoid transfer function:

$$f(x) = (1 - \exp(-2x))/(1 + \exp(-2x)) \tag{9}$$

This function has the following property:

$$f'(x) = (1 - f(x))(1 + f(x)) \tag{10}$$

$$f''(x) = -2f(x)f'(x) \tag{11}$$

In order to decrease the chance of being trapped into local minima, the Hybrid-I algorithm combined with magnified gradient function [21] is proposed in the paper. According to Eq. 10, when $f'\left(\hat{h}_{j_l}^S\right)$ $(l = 1, \ldots, L-1)$ included in Eqs. 7, 8 approaches extreme values (i.e., -1 or 1), $f'\left(\hat{h}_{j_l}^S\right)$ $(l = 1, \ldots, L-1)$ will become so small (close to zero) that $\Delta\omega_{j_l j_{l-1}}$ $(l = 1, \ldots, L-1)$ will approaches zero. So the network will be trapped into a flat region so that it converges more slowly to the global optimal solution or can not converge to the global optimal solution. To overcome this problem, the factors $f'\left(\hat{h}_{j_l}^S\right)$ $(l = 1, \ldots, L-1)$ are magnified in this improved algorithm by using a power factor $1/K$ where the magnified gradient coefficient K is a positive real number greater than or equal to 1 ($K \geq 1$), i.e., to replace $f'\left(\hat{h}_{j_l}^S\right)$ $(l = 1, \ldots, L-1)$ by $\left(f'\left(\hat{h}_{j_l}^S\right)\right)^{1/K}$ $(l = 1, \ldots, L-1)$. When compared with the standard BP algorithm, the gradient term should have a larger increment when $f'\left(\hat{h}_{j_l}^S\right)$ $(l = 1, \ldots, L-1)$ approaches zero so that the network will have lower frequency of being trapped into a flat spot and converge faster to the global optimal solution.

Moreover, since the above modified algorithm incorporates the first-order derivatives of the neural activation at hidden layers into the sum-of-square error cost function, it can reduce the input-to-output mapping sensitivity [8–10].

The modified local search algorithm combines Hybrid-I algorithm with magnified gradient function, and we call it MGFHIA.

4 APSO encoding a priori information from the approximated function

Since a neural network with single nonlinear hidden layer is capable of forming an arbitrarily close approximation of any continuous nonlinear mapping [24–26], our discussion will be limited to the single-hidden layered feedforward neural networks (SLFN).

In the course of approximating a function, the FNN can approximate it more accurately when a priori information containing the function properties is encoded into the network. Sine the first-order derivatives of a function play an important role in the shape of the function, a priori information containing the first-order derivatives of the approximated function is considered in this paper.

Assume that the sample points of the function are selected at identical spaced intervals. In addition, these sample points, i.e.,(x_i, t_i), $i = 1, 2, \ldots, N$, where $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$, $t_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in R^m$, are assumed to be very close in space.

First, a method is presented to obtain the approximation value of the first-order partial derivative of the approximated function. According to Mean-Value theorem, the corresponding approximate estimated values of the functional first-order partial derivative can be obtained as follows:

$$g'(x_{il}) \approx (t_{i+1} - t_{i-1})/(x_{(i+1)l} - x_{(i-1)l}),$$
$$i = 2, \ldots, N-1. \quad l = 1, 2, \ldots, n. \tag{12}$$

$$g'(x_{1l}) \approx (t_2 - t_1)/(x_{2l} - x_{1l}),$$
$$g'(x_{Nl}) \approx (t_N - t_{N-1})\big/(x_{Nl} - x_{(N-1)l}) \quad l = 1, 2, \ldots, n. \tag{13}$$

Obviously, the closer the distances among the sample points are, the more accurate the corresponding approximate estimated values of the functional first-order derivation are.Assume that the SLFN, $\phi(\cdot)$, is used to approximate the function $g(\cdot)$. Then the first-order derivative of the network output with respect to x_{j_1} can be obtained as:

$$\phi'(x_{j_1}) = \left[\sum_{j_2=1}^{H} w_{1,j_2} f'\left(\hat{h}_{j_2}\right) w_{j_2 j_1}, \ldots, \sum_{j_2=1}^{H} w_{m,j_2} f'\left(\hat{h}_{j_2}\right) w_{j_2 j_1}\right]^T \tag{14}$$

where w_{k,j_2} denotes the weights from j_2th hidden neuron to kth output neuron and $w_{j_2 j_1}$ denotes the weights from j_1th input neuron to j_2th hidden neuron. H is the number of the hidden neurons.When the APSO is used to train the above SLFN, each particle represents the weights from the input layer to the hidden layer, or the ones from the hidden layer to the output layer and the corresponding thresholds. In order to encode the first-order information into APSO, a new fitness function is defined as follows:

$$\text{fit} = \frac{1}{N}\sum_{i=1}^{N} \|y_i - t_i\|_2 + \frac{\xi}{N}\sum_{i=1}^{N}\sum_{l=1}^{n} \|g'(x_{il}) - \phi'(x_{il})\|_2 \tag{15}$$

where N is the number of the samples and ξ is the coefficient between 0 and 1. The first term in the right hand of Eq. 15 is the mean sum-of-square error between the target output values and the true ones, and the second term denotes the mean sum-of-square error between the approximate estimated values of first-order partial derivative values for the approximated function and the true ones. Since the new fitness function contains the first-order partial derivative information, the new APSO is referred as APSOAEFDI [20].

In order to overcome the drawbacks from gradient-based learning algorithm for FNN, the local search is combined with the global search in the improved approach. First, the SLFN is trained by APSOAEFDI first to near the global minima. Second, the network is trained again by the modified algorithm—MGFHIA. The MGFHIA penalizes the input-to-output mapping sensitivity of the network in the course of learning. Moreover, because of encoding the priori information and APSOA, the improved approach has better performance than gradient-based learning algorithm. Since the improved approach combines APSOAEFDI and MGFHIA, it is referred to as APSOAEFDI-MFGHIA.

5 Experimental results

In this section, some experiments are conducted to verify the efficiency and effectiveness of our proposed learning approach. All the simulations for BP algorithm, Hybrid-I algorithm, Hybrid-II algorithm, MHLA, APSOA-BP which combines APSOA with BP algorithm, APSOA-HILA which combines APSOA with Hybrid-I algorithm, APSOAEFDI-MHLA and APSOAEFDI-MGFHIA are carried out in MATLAB 6.5 environment running in a Pentium 4, 2.60 GHz CPU.

In the following we shall conduct the experiments with two differentiable functions, i.e., the function $y = (1 - (40x/\pi) + 2(40x/\pi)^2 - 0.4(40x/\pi)^3)e^{-x/2}$ and a sinc function $y = \sin(5x)/(5x)$. The activation functions of the neurons in the hidden layer for eight algorithms all are tangent sigmoid function and the output layers all are linear. The number of the hidden neurons all is 12.

As for the function, $y = (1 - (40x/\pi) + 2(40x/\pi)^2 - 0.4(40x/\pi)^3e^{-x/2}$, assume that the number of the total training data is 126, which are selected from $[o, \pi)$ at identically spaced intervals. A total of 125 testing samples are selected from $[0.0125, \pi - 0.0125)$ at identically spaced intervals. As a result, the approximation errors of the test samples for the improved approach are shown in Fig. 1a.

Similarly, as for the sinc function, assume that 121 training samples are selected from [0, 3] and 120 testing samples are selected from [0.0125, 2.9875]. As a result, the approximation errors of the test samples for the above function with the improved approach are shown in Fig. 1b.

In order to statistically compare the approximation accuracies, standard deviation for mean squared error (MSE) of testing data (SDMSETD) and iterated number for approximating the two functions with the above learning algorithms, we conducted the experiments 50 times for each algorithm, and the corresponding results are summarized in Tables 1 and 2.

From the above results, the conclusions can be drawn as follows:

First, as for each function, the testing errors of the improved learning approach are always less than ones of other learning algorithms except for APSOAEFDI–MHLA. This result rests in the fact that the new approach combines APSOA with the a priori information of the approximated function to search the global optimum before perform local search with MGFHIA.

Second, among all learning algorithms, the learning ones which use APSOA to search global optimum converge at not more than 15,000 epochs, whereas the leaning ones which do not use APSOA converge at 30,000 epochs. Moreover, the improved approach converges even at

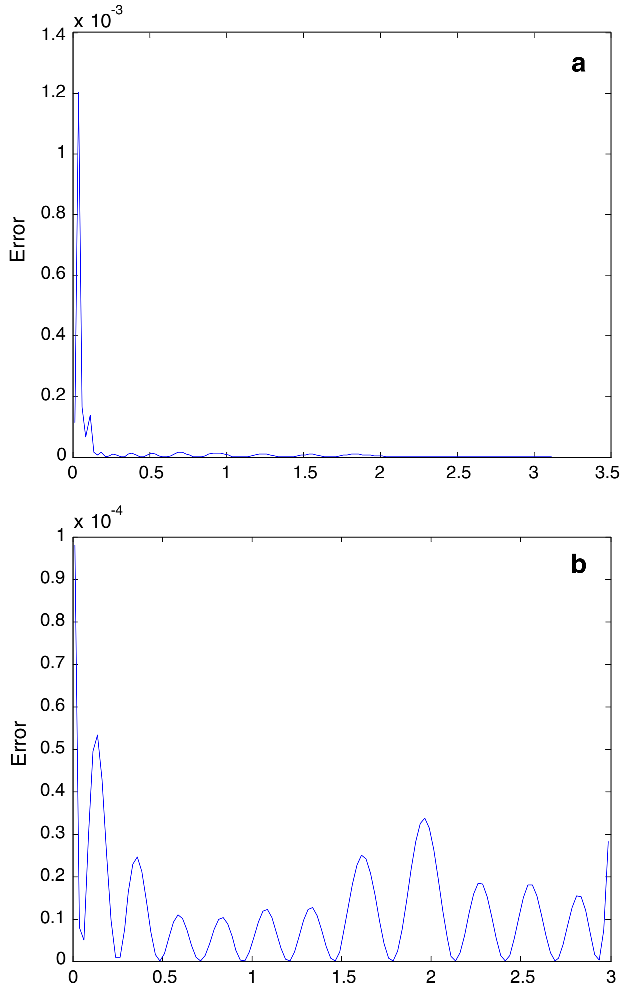

Fig. 1 The curves for the approximation errors of the test samples for two functions with the improved approach. **a** $y = (1 - (40x/\pi) + 2\,(40x/\pi)^2 - 0.4\,(40x/\pi)^3)\,e^{-x/2}$;**b** $y = \sin(5x)/(5x)$

12,000 epochs because of incorporating a priori information and magnified gradient function into SLFN.

Third, the improved approach has slightly worse approximation accuracy than APSOAEFDI–MHLA, while it converges faster than APSOAEFDI–MHLA.

Moreover, in order to verify the efficiency and effectiveness of the proposed learning approach more thoroughly, tenfold cross validation experiments are performed for approximating the above two functions. The corresponding results are show in Tables 3 and 4.

It can be found from Tables 3 and 4 that the values of MSE of the improved learning approach are always less than ones of the other learning ones except for APSOAEFDI–MHLA. This result also supports the above conclusion that the approximation accuracy of the proposed approach is better than the ones of the other learning algorithms but APSOAEFDI–MHLA.

In the following, the corresponding parameters with respect to the improved learning approach for approximating the function $y = (1 - (40x/\pi) + 2\,(40x/\pi)^2 - 0.4\,(40x/\pi)^3)\,e^{-x/2}$ are discussed.

Figure 2 shows the relation between the testing errors and the particle number. It is evident that the testing error is on a downward trend with an increase in the iterated number.

Figure 3 shows the relation between the ultimate testing errors and the temporary testing errors obtained by APSOAEFDI. It can be concluded that the ultimate testing errors have an upward trend as the corresponding temporary testing errors obtained by APSOAEFDI increases.

Figure 4 shows the relation between the ultimate testing errors and the magnified gradient coefficient K in MGFHIA. On the one hand when the magnified gradient coefficient increases from 1 to 1.8, the ultimate testing errors decrease sharply. On the other hand when the magnified gradient coefficient increases from 1.8 to 3, the ultimate testing errors are on an upward trend. This shows that the ultimate testing errors may not get less as the magnified gradient coefficient gets bigger.

6 Conclusions

In this paper, an improved approach for function approximation problem is proposed to obtain better approximation

Table 1 The average values of MSE, the standard deviation for MSE of testing data and iterated number for approximating the function $y = (1 - (40x/\pi) + 2\,(40x/\pi)^2 - 0.4\,(40x/\pi)^3)\,e^{-x/2}$ with eight learning algorithms

Learning algorithms	Training MSE	Testing MSE	SDMSETD	Iterated number
BP	5.2511e-4	4.5036e-4	2.9894e-5	30,000
Hybrid-I	2.6711e-4	2.1255e-4	2.6883e-4	30,000
Hybrid-II	3.0045e-4	2.8659e-4	7.9834e-5	30,000
MHLA	1.5123e-4	1.2102e-4	4.1634e-5	30,000
APSOA–BP	2.6549e-4	1.6577e-4	6.1377e-5	15,000
APSOA–HILA	1.8540e-4	7.9857e-5	7.3256e-5	15,000
APSOAEFDI–MHLA	4.6037e-5	2.3812e-5	2.3370e-5	15,000
APSOAEFDI–MGFHIA	8.6025e-5	4.8657e-5	7.8156e-5	12,000

Table 2 The average values of MSE, the standard deviation for MSE of testing data and iterated number for approximating the sinc function $y = \sin(5x)/(5x)$ with eight learning algorithms

Learning algorithms	Training MSE	Testing MSE	SDMSETD	Iterated number
BP	1.5324e-4	1.4652e-4	8.1885e-4	30,000
Hybrid-I	7.9048e-5	7.8599e-5	4.5364e-4	30,000
Hybrid-II	8.3157e-5	7.6853e-5	4.1619e-4	30,000
MHLA	4.8696e-5	4.4998e-5	4.3326e-4	30,000
APSOA–BP	7.6217e-5	7.5632e-5	6.5365e-4	15,000
APSOA–HILA	4.9942e-5	4.7612e-5	5.1238e-4	15,000
APSOAEFDI–MHLA	2.0874e-5	1.8991e-5	3.6174e-4	15,000
APSOAEFDI–MGFHIA	4.1914e-5	3.0760e-5	4.8623e-4	12,000

Table 3 The MSE of approximating the function $y = (1 - (40x/\pi) + 2\,(40x/\pi)^2 - 0.4\,(40x/\pi)^3)\,e^{-x/2}$ for 20 times by tenfold cross-validation with eight algorithms

Learning algorithms	Training MSE	Testing MSE
BP	0.0013	0.0183
Hybrid-I	6.2749e-4	0.0059
Hybrid-II	8.0919e-4	0.0058
MHLA	6.7420e-4	0.0056
APSOA–BP	7.2315e-4	0.0065
APSOA–HILA	6.0124e-4	0.0054
APSOAEFDI–MHLA	5.0365e-4	0.0030
APSOAEFDI–MGFHIA	4.0365e-4	0.0032

Table 4 The MSE of approximating the function $y = \sin(5x)/(5x)$ for 20 times by tenfold cross-validation with eight algorithms

Learning algorithms	Training MSE	Testing MSE
BP	0.0019	0.1410
Hybrid-I	0.0038	0.0065
Hybrid-II	0.0045	0.0073
MHLA	4.1563e-4	0.0043
APSOA–BP	5.2265e-4	0.0068
APSOA–HILA	6.4563e-4	0.0059
APSOAEFDI–MHLA	0.0010	6.1186e-4
APSOAEFDI–MGFHIA	2.1949e-4	0.0038

accuracy and faster convergence rate. In the improved approach, global search algorithm is combined with local search algorithm reasonably. First, the network is trained to search global minima by encoding the first-order derivative information of the approximated function into APSO. Second, with the trained weights produced by APSO, the SLFN is trained by the modified gradient-based local search algorithm with magnified gradient function. Moreover, the modified local search algorithm penalizes the input-to-output mapping sensitivity of network and avoids

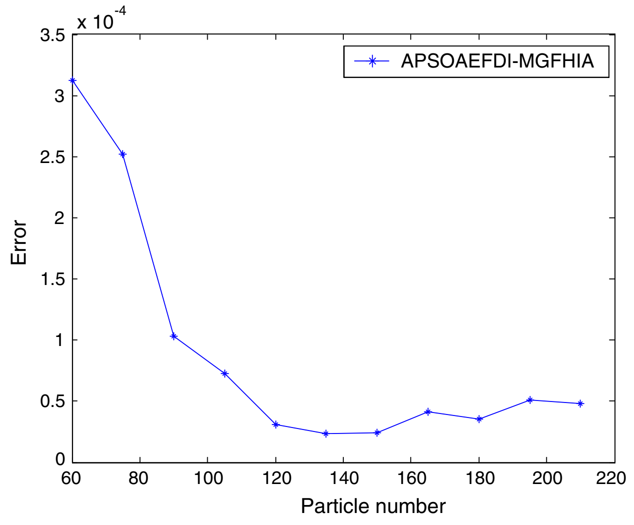

Fig. 2 The relations between the testing errors and the particle number with the improved learning approach for approximating the function $y = (1 - (40x/\pi) + 2\,(40x/\pi)^2 - 0.4\,(40x/\pi)^3)\,e^{-x/2}$

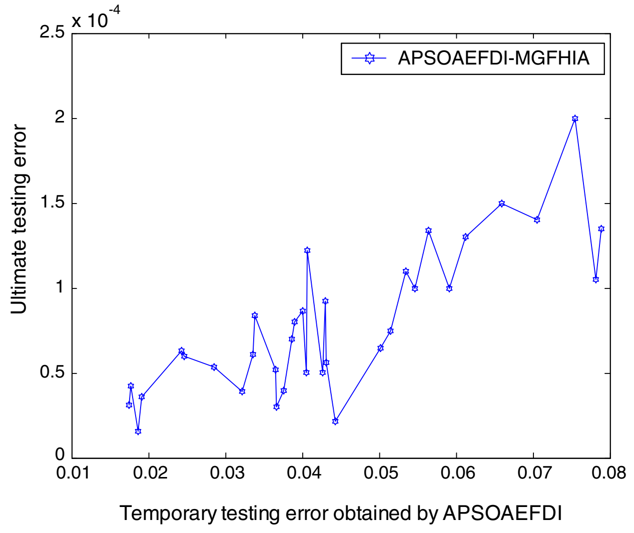

Fig. 3 The relations between the ultimate testing errors and the temporary testing errors obtained by APSOAEFDI with the improved learning approach for approximating the function $y = (1 - (40x/\pi) + 2\,(40x/\pi)^2 - 0.4\,(40x/\pi)^3)\,e^{-x/2}$

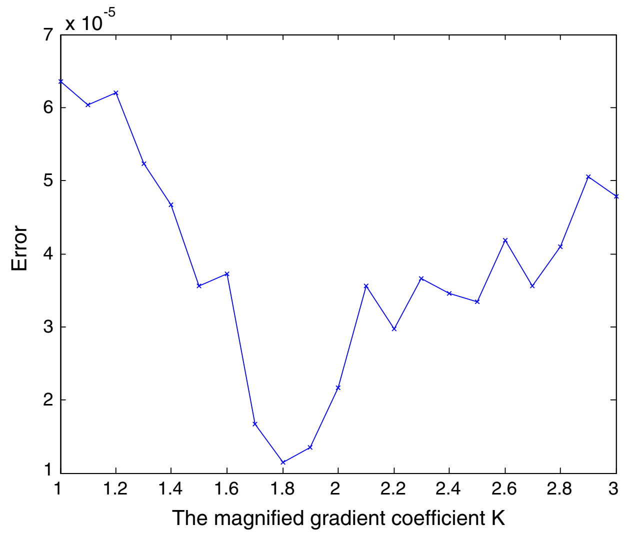

Fig. 4 The relations between the ultimate testing errors and the magnified gradient coefficient in MGFHIA for approximating the function $y = (1 - (40x/\pi) + 2\,(40x/\pi)^2 - 0.4\,(40x/\pi)^3)\,e^{-x/2}$

being trapped into local minima in the course of learning. Due to combined APSO and the a priori information with the local search algorithm, the improved approach has better approximation accuracy and convergence rate. Finally, simulation results are given to verify the efficiency and effectiveness of the proposed learning approach. Future research works will include how to apply the proposed learning algorithm to resolve more numerical computation problems.

Acknowledgments This work was supported by the National Science Foundation of China (No. 60702056) and the Initial Funding of Science Research of Jiangsu University (No. 07JDG033).

References

1. Nasr MB, Chtourou M (2009) A fuzzy neighborhood-based training algorithm for feedforward neural networks. Neural Comput Appl 18(2):127–133. doi:10.1007/s00521-007-0165-z
2. Huang DS (2004) A constructive approach for finding arbitrary roots of polynomials by neural networks. IEEE Trans Neural Netw 15:477–491. doi:10.1109/TNN.2004.824424
3. Huang DS, Chi ZR (2004) Finding roots of arbitrary high order polynomials based on neural network recursive partitioning method. Sci China Ser Inf Sci 47:232–245
4. Huang DS, Horace Ip HS, Chi ZR (2004) A neural root finder of polynomials based on root momnets. Neural Comput 16:1721–1762. doi:10.1162/089976604774201668
5. Huang DS, Horace HS Ip, Chi ZR, Wong HS (2003) Dilation method for finding close roots of polynomials based on constrained learning neural networks. Phys Lett A 309:443–451. doi:10.1016/S0375-9601(03)00216-0
6. Han F, Huang DS (2008) A new constrained learning algorithm for function approximation by encoding a priori information into feedforward neural networks. Neural Comput Appl 17(5–6):433–439
7. Li SG, Wu ZM (2008) Business performance forecasting of convenience store based on enhanced fuzzy neural network. Neural Comput Appl 17(5–6):569–578
8. Jeong SY, Lee SY (2000) Adaptive learning algorithms to incorporate additional functional constraints into neural networks. Neurocomputing 35:73–90. doi:10.1016/S0925-2312(00)00296-4
9. Jeong DG, Lee SY (1996) Merging back-propagation and Hebbian learning rules for robust classifications. Neural Netw 9:1213–1222. doi:10.1016/0893-6080(96)00042-1
10. Han F, Huang DS, Cheung YM, Huang GB (2005) A new modified hybrid learning algorithm for feedforward neural networks, vol 3496. In: International symposium on neural network, Chongqing, 30 May–1 June, China. Lecture Notes in Computer Science, Springer, Berlin, pp 572–577
11. Eberhart RC, Kennedy J (1995) A new optimizer using particles swarm theory. In: Proceeding of sixth international symposium on micro machine and human science, Nagoya, Japan, pp 39–43
12. Eberhart RC, Kennedy J (1995) Particle swarm optimization, proceeding of IEEE International Conference on Neural Network, Perth, Australia, pp 1942–1948
13. Parrott D, Li XD (2006) Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Trans Evol Comput 10(4):440–458. doi:10.1109/TEVC.2005.859468
14. Blackwell T, Branke J (2006) Multiswarms, exclusion, and anti-convergence in dynamic environments. IEEE Trans Evol Comput 10(4):459–472. doi:10.1109/TEVC.2005.857074
15. Clerc M, Kennedy J (2002) The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6(1):58–73. doi:10.1109/4235.985692
16. Parsopoulos K, Vrahatis M (2002) Recent approaches to global optimization problems through particle swarm optimization. Nat Comput 1(2–3):235–306. doi:10.1023/A:1016568309421
17. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading
18. Langdon WB, Poli R (2007) Evolving problems to learn about particle swarm optimizers and other search algorithms. IEEE Trans Evol Comput 11(5):561–578. doi:10.1109/TEVC.2006.886448
19. Wang YP, Dang CY (2007) An evolutionary algorithm for global optimization based on level-set evolution and latin squares. IEEE Trans Evol Comput 11(5):579–595. doi:10.1109/TEVC.2006.886802
20. Han F, Ling QH (2008) A new approach for function approximation incorporating adaptive particle swarm optimization and a priori information. Appl Math Comput 205(2):792–798. doi:10.1016/j.amc.2008.05.025
21. Ng SC, Cheung CC, Leung SH (2004) Magnified gradient function with deterministic weight modification in adaptive learning. IEEE Trans Neural Netw 15(6):1411–1423. doi:10.1109/TNN.2004.836237
22. Shi YH, Eberhart RC (1998) A modified particle swarm optimizer. In: Proceedings of IEEE world conference on computation intelligence, pp 69–73
23. Shi YH, Eberhart RC (1998) Parameter selection in particle swarm optimization. In: 1998 annual conference on evolutionary programming, San Diego, March
24. Funahashi K (1989) On the approximate realization of continuous mapping by neural networks. Neural Netw 2(3):183–192. doi:10.1016/0893-6080(89)90003-8
25. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal, approximators. Neural Netw 2(5):359–366. doi:10.1016/0893-6080(89)90020-8
26. Irie B, Miyake S (1988) Capabilities of three-layered perceptions. In: Proceedings of the IEEE conference on neural networks, vol I. San Diego, CA, pp 641–648