INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 10 (11): 2226-2231, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.2226.2231
© 2011 Asian Network for Scientific Information

An Improved PSO Algorithm Coupling with Prior Information for Function Approximation

[1,2]Juanjuan Tu, [1]Yongzhao Zhan and [1]Fei Han
[1]School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang, Jiangsu 212013, China
[2]School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang, Jiangsu 212003, China

Abstract: An improved Particle Swarm Optimization (PSO) algorithm coupling with prior information for function approximation is proposed in present study. The prior information derived from the first-order derivative information of the approximated function is used to adjust the position of the particles. In the new approximation algorithm, feedforward neural network is first trained by improved PSO and then by BP. The prior information narrows the search space and guides the movement direction of the particles, so the convergence rate and the generalization performance are improved. Experimental results demonstrate that the new algorithm is more effective than traditional methods.

Key words: Feedforward neural network, particle swarm optimization, prior information, function approximation, BP neural network

INTRODUCTION

Feedforward Neural Network (FNN) is known to be universal approximator for any non-linear function (Lawrence *et al*., 1996; Meng and Sun, 2000). The learning algorithm determines the rules for optimizing parameters of network within the training period. Backpropagation (BP) is one of the most used algorithm for FNN learning. It adjusts the weights and thresholds of network based on the gradient information of the fitness function in order to reduce the errors over all input patterns (Werbos, 1990). However, gradient methods may only converge to a local optimum and are sensitive to the values of initial parameters (Rumelhart *et al*., 1986). Many improved BP algorithms have also these shortcomings in essence.

In recent years, some swarm intelligent algorithms such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) have been used to optimize FNN because of their good capability of global search. The error functions are less likely to be trapped in a local optimum and need not be differentiable or even continuous (Sexton and Dorsey, 2000; Parrott and Xiaodong, 2006). Compared with GA, PSO has some advantages. First, the computing formula is simple, whereas, GA has complex genetic computations such as crossover and mutation. Moreover, PSO has memory. Finally, PSO converges more rapidly than GA in most cases (Yang *et al*., 2007). FNN optimized with PSO is more suitable for function approximation.

The basic PSO algorithm has some drawbacks. It is easy to lose diversity and cannot converge to global optimum with probability 1 (Lovbjerg and Krink, 2002; Ozcan and Mohan, 1999). In order to overcome these drawbacks, many improved PSO algorithms have been proposed from different aspects, such as the Adaptive Particle Swarm Optimization (APSO) (Shi and Eberhart, 1998) and the Quantum-behaved Particle Swarm Optimization (QPSO) (Sun *et al*., 2004) etc. They have been effective to some extent but sometimes may lose convergence direction.

In fact, the prior information abstracted from samples can be used to speed up convergence rate and improve generalization performance (Lv *et al*., 2004). In the literature (Han and Ling, 2008) an approach for function approximation based on APSO and a prior information (APSOAEFDI-MHLA) was proposed which the fitness function was modified according to the derivative information. In this study, a new PSO algorithm coupling with prior information is presented. After the position of each particle is initialized randomly, the prior information derived from the first-order derivative of the function is used to adjust the position of the particles. As the prior information guides the direction of particle flight, the

Corresponding Author: Juanjuan Tu, School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang, Jiangsu 212013, China

search space of PSO is narrowed and particle swarm can converge to global optimum more quickly. For function approximation, FNN is trained by the improved PSO first and then by local search algorithm-BP.

PARTICLE SWARM OPTIMIZATION

PSO is a kind of algorithm to search for the best solution by simulating the movement and flocking of birds. It is a form of evolutionary computation technique firstly described by Kennedy and Eberhart (1995). The basic PSO formula defines each particle as a potential solution to a problem in D-dimensional space, with particle i denoted as $X_i = (x_{i1}, x_{i2}, x_{iD})$. Each particle maintains a memory of its previous best position and a velocity along each dimension, represented as $P_i = (P_{i1}, P_{i2}, P_{iD})$. At each iteration, the P vector of the particle with the best fitness in the global neighborhood, designated g and the P vector of the current particle are combined to adjust the velocity along each dimension and that velocity is then used to compute a new position for the particle.

In the PSO algorithm of Kennedy and Eberhart, the formula of the velocity and position of particle i are updated as following:

$$V_{id} = V_{id} + c_1 * r \text{ and } ()(P_{id} - X_{id}) + c_2 * r \text{ and } () * (P_{gd} - X_{gd}) \quad (1)$$

$$X_{id} = X_{id} + V_{id} \quad (2)$$

where, i = 1, 2, ..., N, d = 1, 2, ..., D and N is the swarm size. c_1 and c_2 are called learning rates which determine the relative influence of the social and cognition components. Furthermore, a constant-Vmax is used to arbitrarily limit the velocities of the particles and improve the resolution of the search.

PSO is widely used to train FNN and its basic idea of is that PSO replaces traditional learning algorithm such as BP to train parameters of FNN (Salerno, 1997; Settles and Rylander, 2002). Each particle is a vector and represents a set of parameters. In other words, the parameters are evolved every iteration for each particle.

The training error is used to compute the fitness f (x):

$$f(x) = \frac{1}{1 + \frac{1}{2n}\sum_{p=1}^{n}(y_p - t_p)^2} \quad (3)$$

where, p is the number of samples, y_p is actual output and t_p is given output.

After either the maximum number of iterations or the minimum error cutoff has been reached, the program terminates and the optimum solution is obtained from particle swarm. This optimum solution is the particle with the best fitness--the one with the lowest error.

PSO COUPLING WITH PRIOR INFORMATION FOR FUNCTION APPROXIMATION

PSO coupling with prior information (PI-PSO) can improve convergence rate. For function approximation, many function properties abstracted from samples can be used to derive prior information. In present study, the first--order derivative of function is considered for three layers FNN and the output layer has only one node.

Before introduction the computing process of prior information, we first make the following mathematical notation about samples. Assume that (x_i, t_i) represents the i th sample, where i = 1, 2,..., N, $x_i = [x_{i1}, x_{ij1},..., x_{in}]^T \epsilon R^n$.

The first-order derivative of the network output with respect to x_{j1} can be obtained as:

$$g'(x_{j1}) = \sum_{j_2=1}^{H} w_{j2} * f'(\hat{h}_{j2}) * w_{j2j1} \quad (4)$$

where:

$$f(\hat{h}_{j2}) = \frac{1}{1 + e^{-\lambda u}}$$

and:

$$u = \sum_{j1=1}^{n} w_{j2j1} * x_{ij1} + \theta_{j2}$$

(Lv *et al*., 2004). w_{j2} denotes the weight from the j^2 th hidden neuron to the output neuron and w_{j2j1} denotes the weight from the j^1 th input neuron to the j^2 th hidden neuron. θ_{j2} denotes the threshold of the j^2 th hidden neuron and H is the number of the hidden neurons.

According to Mean-Value theorem, the approximate estimated values of the functional first-order partial derivative can be obtained as (Lv *et al*., 2004):

$$f'(x_{ij1}) = \frac{t_{(i+1)} - t_{(i-1)}}{x_{(i+1)j1} - x_{(i-1)j1}} \quad (5)$$

Assume that $g'(x_{j1}) = f'(x_{ij1})$, then the following expression can be derived:

$$\sum_{j2=1}^{H} w_{j2} * w_{j2j1} * [\sum_{j1=1}^{n} w_{j2j1} * \frac{\lambda * e^{-\lambda u}}{(1+e^{-\lambda u})^2}] = \frac{t_{(i+1)} - t_{(i-1)}}{x_{(i+1)j1} - x_{(i-1)j1}} \qquad (6)$$

Obviously, Eq. 6 shows the relation between w_{j2} and w_{j2j1}. In PSO-FNN, the relation can be used as important prior information.

In PSO algorithm, after the initial position of each particle is produced randomly, the value of corresponding dimensions would be modified according to above relation because each particle represents a set of parameters. w_{j2} will increase or decrease dynamically until the value of $/g'(x_{j1})$-$f'(x_{ij1})/$ is less than ξ, where ξ is a very little value. For example, w_{j1} may add or subtract $\sqrt{/g'(x_{j1}) - f'(x_{ij1})/}$. In iterative process, p_{gd} would be modified by similar operation to guide each particle's flight. Thus the searching space is narrowed and the global optimum can be obtained more quickly.

PSO has good capability of global search but BP is more efficient in local search. In FNN training process, PSO combined with BP algorithm can be used to achieve optimum, effectively. The neural network is trained by PSO to near the global optimum and then is trained by BP. The algorithm is called PSO-BP.

In this study, we use PSO coupling with prior information and BP to train FNN. The algorithm is referred to as PI-PSO-BP.

EXPERIMENTAL RESULTS

In order to demonstrate the good performance of the new approach, a lot of experiments have been conducted in MATLAB 7.0. The simulations for function approximation of six algorithms which are BP, PSO-BP, APSO-BP, QPSO-BP, APSOAEFDI-MHLA and PI-PSO-BP have been carried out. The following two functions are selected:

$$y = (1 - (40x/\pi) + 2(40x/\pi)^2 - 0.4(40x/\pi)^3)e^{-20x/\pi} \qquad (7)$$

$$y = \sin(5x)/(5x) \qquad (8)$$

As for function (7), assume that the number of the total training data is 126 which are selected from $[0, \pi]$ at identically spaced intervals. 125 testing samples are selected from $[0.0125, \pi\text{-}0.0125]$ at identically spaced intervals. Similarly, as for function (8), assume 121 training samples are selected from [0, 3] at identically spaced intervals. 120 testing samples are selected from [0.0125, 2.9875] at identically spaced intervals.

The approximating results of the PI-PSO-BP algorithm for the above two functions are shown in Fig. 1a and b.

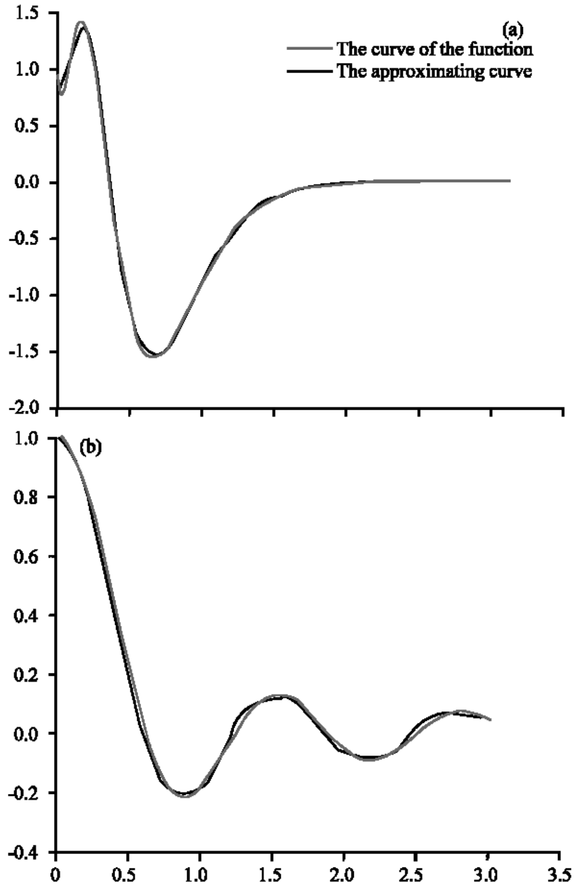

Fig. 1 (a-b): The approximating results of the PI-PSO-BP algorithm for approximating function: (a) $y = (1\text{-}(40x/\pi) + 2(40x/\pi)^2 \text{-}0.4(40x/\pi)^3)e^{-20x/\pi}$ (b) $y = \sin(5x)/(5x)$

Table 1: The average values of MSE and iteration No. for approximating the function $y = (1\text{-}(40x/\pi) + 2(40x/\pi)^2 \text{-}0.4(40x/\pi)^3) e^{-20x/\pi}$ with six learning algorithms

Learning algorithms	Training MSE	Testing MSE	Iteration No.
BP	5.3561e-4	4.6163e-4	30000
PSO-BP	3.0835e-4	2.2546e-4	15000
APSO-BP	2.6549e-4	1.6577e-4	15000
QPSO-BP	1.1752e-4	9.7861e-5	15000
APSOAEFDI-MHLA	4.6037e-5	2.3812e-5	15000
PI-PSO-BP	5.4628e-5	3.6791e-5	15000

The testing errors of the PI-PSO-BP algorithm for the two functions are shown in Fig. 2a and b.

Mean Squared Error (MSE) and iteration number are used to compare the performance of each algorithm. The experiments are conducted for fifty times and the corresponding results are summarized in Table 1 and 2.

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

First, as for each function, the iteration numbers of BP are more than the other FNN learning algorithms which

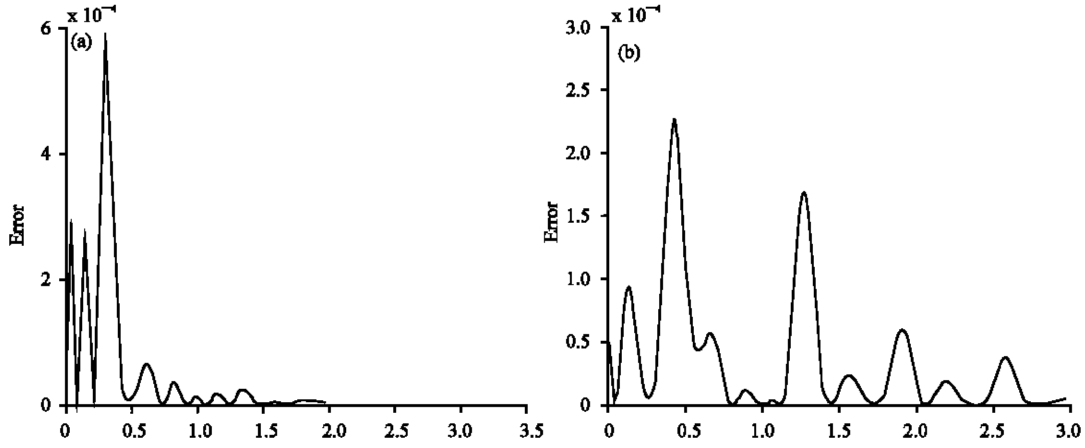

Fig. 2 (a-b): The testing errors of the PI-PSO-BP algorithm for approximating function: (a) $y = (1-(40x/\pi) + 2\ (40x/\pi)^2 -0.4\ (40x/\pi)^3)\ e^{-20x/\pi}$; (b) y = sin (5x)/(5x)

Table 2: The average values of MSE and iteration number for approximating the function y = sin (5x)/(5x) with six learning algorithms

Learning algorithms	Training MSE	Testing MSE	Iteration No.
BP	1.5324e-4	1.4652e-4	30000
PSO-BP	8.2691e-5	7.9532e-5	15000
APSO-BP	7.6217e-5	7.5632e-5	15000
QPSO-BP	6.8073e-5	6.6594e-5	15000
APSOAEFDI-MHLA	2.0874e-5	1.8991e-5	15000
PI-PSO-BP	3.1562e-5	2.7675e-5	15000

use PSO or improved PSO. The results show that PSO can speed up the training process of FNN by its capability of global search.

Second, the testing errors of PI-PSO-BP are close to APSOAEFDI-MHLA and are always less than the other four algorithms. The results demonstrate that PSO coupling with prior information can improve the approximating accuracy of FNN effectively.

Finally, Fig. 1 shows that the approximating curves are almost consistent with the curves of the functions. The new approach for function approximation works well. In the following, the corresponding parameters with respect to the new learning approach for approximating the two functions are discussed. The experimental results are shown from Fig. 3 to 5.

From Fig. 3 to 5, the conclusions can be drawn as follows:

First, Fig. 3 shows that as for function (b), the testing error has a downward trend when the number of hidden neurons is less than 8 and then rises again. For function (a), the testing error is the lowest when the number of hidden neurons is 10.

Second, Fig. 4 shows the relation between the testing error and the number of particles. Obviously, the most suitable number is 30 for the two functions.

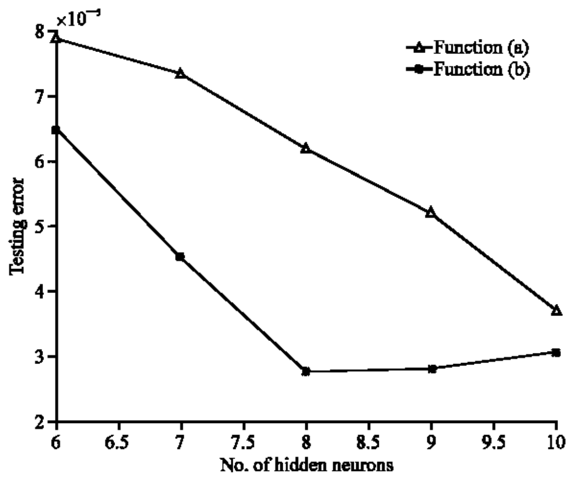

Fig. 3: The relationship between the number of hidden neurons and testing error for approximating the functions: (a) $y = (1-(40x/\pi) + 2\ (40x/\pi)^2 -0.4\ (40x/\pi)^3)\ e^{-20x/\pi}$; (b) y = sin (5x)/(5x)

Finally, it can be found from Fig. 5 that the more the numbers of iteration are, the less the testing errors are.

In addition, the relationship between the iteration numbers of PSO and BP in three algorithms for approximating the two functions is listed in Table 3 and 4.

From Table 3 and 4, we can find that the testing errors of the new learning approach are still the lowest one when the iteration numbers of PSO and BP are all less than the other two algorithms. The results show that the prior information helps PSO converge faster and BP can find more accurate global optimum in shorter time. So, the generalization performance of PI-PSO-BP is the best.

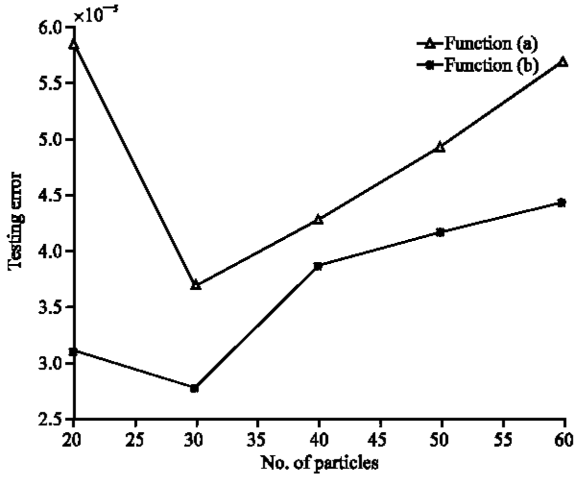

Fig. 4: The relationship between the number of particles and testing error for approximating the functions: (a) $y = (1-(40x/\pi) + 2\,(40x/\pi)^2 -0.4\,(40x/\pi)^3)\ e^{-20x/\pi}$; (b) $y = \sin(5x)/(5x)$

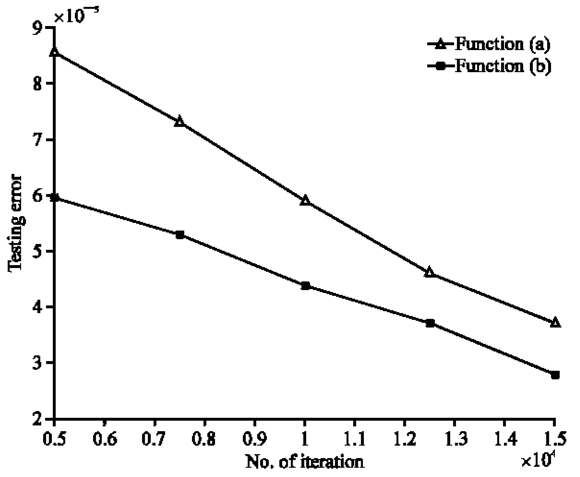

Fig. 5: The relationship between the number of iteration and testing error for approximating the functions: (a) $y = (1-(40x/\pi) + 2\,(40x/\pi)^2 -0.4\,(40x/\pi)^3)\ e^{-20x/\pi}$; (b) $y = \sin(5x)/(5x)$

Table 3: The relationship between the iteration numbers of PSO and BP in three algorithms for approximating the function: $y = (1-(40x/\pi) + 2(40x/\pi)^2 -0.4(40x/\pi)^3)\ e^{-20x/\pi}$

Learning algorithms	Iteration No. of PSO	Iteration No. of BP	Testing MSE
PSO-BP	150	15000	2.2546e-4
QPSO-BP	150	15000	9.7861e-5
PI-PSO-BP	100	10000	5.8742e-5

Table 4: The relationship between the iteration numbers of PSO and BP in three algorithms for approximating the function: $y = \sin(5x)/(5x)$

Learning algorithms	Iteration No. of PSO	Iteration No. of BP	Testing MSE
PSO-BP	150	15000	7.9532e-5
QPSO-BP	150	15000	6.6594e-5
PI-PSO-BP	100	10000	4.3601e-5

CONCLUSIONS

Present study, an improved PSO algorithm is proposed. According to the prior information derived from the first-order derivative of function, the corresponding particle positions are modified. Then the new algorithm combined with BP is used to train FNN for function approximation. The experimental results show that PSO coupling with prior information has better convergence rate and can improve approximating accuracy. Future research works will find more effective prior information for PSO algorithm.

ACKNOWLEDGMENTS

This study was supported by the National Natural Science Foundation of China (Nos. 60702056, 61003183) and the Natural Science Foundation of Jiangsu Province (Nos. BK2009197, BK2009199).

REFERENCES

Han, F. and Q. Ling, 2008. A new approach for function approximation incorporating adaptive particle swarm optimization and a priori information. Applied Math. Comput., 205: 792-798.

Kennedy, J. and R.C. Eberhart, 1995. Particle swarm optimization. Proc. IEEE Int. Conf., 27: 1942-1948.

Lawrence, S., A.C. Tsoi and A.D. Back, 1996. Function approximation with neural networks and local methods: Bias, variance and smoothness. Proceedings of the Australian Conference on Neural Network (ACNN'96), Canberra, Australia, pp: 16-21.

Lovbjerg, M. and T. Krink, 2002. Extending particle swarm optimizers with self-organized critically. Proceedings of the IEEE International Conference On Evolutionary Computation (ICEC'02), Honolulu, HI, USA, pp: 1588-1593.

Lv, B., C. Chunyi and P. Hongtailang, 2004. The new learning method of prior information by using three-layers neuron network. Chinese Sci., 34: 374-390.

Meng, J. and Z. Sun, 2000. Application of combined neural networks in nonlinear function approximation. Proceedings of the 3rd World Congress on Intelligent Control and Automation (WCICA'00), Hefei, China, pp: 839-841.

Ozcan, E. and C. Mohan, 1999. Particle swarm optimization: Surfing the waves. Proceedings of the Congress on Evolutionary Computation (CEC'99), Washington, DC., USA., pp: 1939-1944.

Parrott, D. and L. Xiaodong, 2006. Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Trans. Evol. Comput., 10: 440-458.

Rumelhart, D.E., G.E. Hinton and R.J. Williams, 1986. Learning representations by back propagating errors. Nature, 323: 533-536.

Salerno, J., 1997. Using the particle swarm optimization technique to train a recurrent neural model. Proceedings of IEEE International Conference on Tools with Artificial Intelligence, Nov. 3-8, Newport Beach, CA., USA., pp: 45-49.

Settles, M. and B. Rylander, 2002. Neural network learning using particle swarm optimizers. Advances in Information Science and Soft Computing, pp: 224-226.

Sexton, R.S. and R.E. Dorsey, 2000. Reliable classification using neural networks: A genetic algorithm and backpropagation comparison. Decis. Support Syst., 30: 11-22.

Shi, Y. and R. Eberhart, 1998. A modified particle swarm optimizer. Proceedings of the IEEE Congress on Evolutionary Computation, May 4-9, Piscataway, New Jersey, pp: 69-73.

Sun, J., B. Feng and W. Xu, 2004. Particle Swarm optimization with particles having quantum behavior. Proceedings of Congress on Evolutionary Computation, June 19-23, Harbin, pp: 325-331.

Werbos, P.J., 1990. Backpropagation through time: What it does and how to do it. Proc. IEEE, 78: 1550-1560.

Yang, X.M., J.S. Yuan, J.Y. Yuan and H.N. Mao, 2007. A modified particle swarm optimizer with dynamic adaptation. Applied Math. Comput., 189: 1205-1213.