International Journal of Hybrid Information Technology
Vol.9, No.3 (2016), pp. 347-354
http://dx.doi.org/10.14257/ijhit.2016.9.3.33

A Novel Neural Network Algorithm Optimized by PSO for Function Approximation

Juanjuan Tu[1,a],*, Wenlan Zhou[2,b] and HongmeiLi[1,c]

[1]*School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang, Jiangsu 212003, China*
[2]*School of electronic information, Jiangsu University of Science and Technology, Zhenjiang, Jiangsu212003, China*
[a,*]*ecsitu@126.com,*[b]*wenlan_zhou@163.com,*[c]*liredmei@yeah.net*

Abstract

A novel neural network algorithm optimized by particle swarm optimization (PSO) for function approximation is proposed in this paper. The prior information extracted from the upper and lower bound of the approximated function is coupled into PSO. Since the prior information narrows the search space and guides the movement direction of the particles, the convergence rate and the approximation accuracy are improved. Experimental results demonstrate that the new algorithm is more effective than traditional methods.

Keywords*: particle swarm optimization, prior information, function approximation*

1. Introduction

Neural network (NN) trained by traditional algorithms such as back-propagation (BP) was used to approximate function in early years [1, 2]. Yet the approximation accuracy is not high because BP has some drawbacks. First, it is easy to fall into local optimum. Second, it converges slowly. Finally, it is sensitive to the values of initial parameters. Many improved BP algorithms are still gradient-based learning methods, so they have also these shortcomings in essence.

In order to overcome the above drawbacks, some swarm intelligent algorithms such as particle swarm optimization (PSO) have been used to optimize the neural network because of their good capability of global search [3, 4]. But the basic PSO algorithm is not the best. For example, it is easy to lose diversity and cannot converge to global optimum with probability 1 [5, 6]. Many improved algorithms have been proposed to enhance the efficiency of PSO, such as the adaptive particle swarm optimization (APSO) [7] and the quantum-behaved particle swarm optimization (QPSO) [8] etc. They have been effective to some extent, but sometimes may lose search direction.

For function approximation, some features of function can be used as prior information. A lot of researchers coupled them into the traditional training algorithms of neural network and acquired certain effect [9-11]. In the literatures [12, 13], two improved algorithms were proposed in which the first-order derivative was abstracted and coupled into PSO. They are APSOAEFDI-MHLA and FOD-PSO-BPNN respectively. Because PSO is a kind of global search algorithm, they are more effective than the local search algorithms coupling with prior information.

In this paper, a new PSO algorithm coupling with prior information extracted from the upper and lower bound (ULB) is presented. After the position of each particle is initialized randomly, the prior information is used to adjust the position of all particles. Neural network is trained by the improved PSO first and then by BP, so the algorithm is named ULB-PSO-BPNN.

ISSN: 1738-9968 IJHIT

This paper is organized as follows: In Section 2, a brief over view of neural network optimized by PSO is given. Section 3 explains how the prior information is coupled into PSO to train neural network. In Section 4, the experimental results of seven algorithms for function approximation aregiven and their performances are compared and contrasted. Conclusionsare drawn in Section 5.

2. Neural Network Optimized by PSO

PSO is firstly described by Kennedy and Eberhart [14]. It simulates the movement and flocking of birds. Each particlehas *D*-dimension and maintains a memory of its previous best position represented as $\mathbf{P}_i=(\mathbf{P}_{i1},\mathbf{P}_{i2},...,\mathbf{P}_{iD})$, where *i* is the number of particle. 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.

The formula of the velocity *V* and position *X* of particle i are updated as following:

$$V_{id}=V_{id}+c_1*rand()(P_{id}-X_{id})+c_2*rand()*(P_{gd}-X_{gd}), \quad (1)$$

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

Where $i=1,2,...,N$, $d=1,2,...,D$ and N is the swarm size. c_1 and c_2 are called learning rates.

PSO can be used to train neural network combined with BP and the algorithm is referred to as PSO-BPNN. Each particle is a vector and represents a set of parameters. The training error is used to compute the fitness $f(x)$:

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

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.

3. ULB-PSO-BPNN for Function Approximation

3.1 PSO-BPNN Coupling with ULB Prior Information

This section explains how the ULB prior information is coupled into PSO.The neural network trained by improved PSO has three layers and the output layer has only one node.

Before introduce the computing process of prior information, we first make the following mathematical notation about samples. Assume that $X(i)$ represents the input value of the*i*th sample and $X(j)$ represents the input value of the *j*th sample,

Where $i=1,2,...N, X(i)=[x_{i1},...x_{ij_1},...,x_{in}]^T\in R^n$.

$f(X)$ is the function to beapproximated. If the output value $y(X)$ is totally close to $f(X)$, formula (4) can be obtained：

$$f(X(i))-f(X(j))=\sum_{j2=1}^{H}w_{j2}\frac{1}{1+e^{-\lambda\mu_i}}-\sum_{j2=1}^{H}w_{j2}\frac{1}{1+e^{-\lambda\mu_j}}, \quad (4)$$

Where $\mu_i = \sum_{j_1=1}^{n} w_{j_2 j_1} * x_{ij_1} + \theta_{j_2}$ and $\mu_j = \sum_{j_1=1}^{n} w_{j_2 j_1} * x_{jj_1} + \theta_{j_2}$. w_{j2} denotes the weight from the $j2$ th hidden neuron to the output neuron and w_{j2j1} denotes the weight from the $j1$ th input neuron to the $j2$ th hidden neuron. θ_{j2} denotes the threshold of the $j2$ th hidden neuron and H is the number of the hidden neurons.

Obviously, formula (4) shows the relation between w_{j2} and w_{j2j1}. The relation can be used as an important prior information. In PSO-BPNN, 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. In iterative process P_{gd} would be modified to guide each particle's flight. Thus the searching space is narrowed and the global optimum can be obtained more quickly.

The step of modification is:

$$r = \sum_{j2=1}^{H} w_{j2} \frac{1}{1+e^{-\lambda\mu_i}} - \sum_{j2=1}^{H} w_{j2} \frac{1}{1+e^{-\lambda\mu_j}};$$

$$s = f(X(i)) - f(X(j));$$

while $(abs(r - s) > \varepsilon)$

{ $t = sqrt(abs(r - s))$;
  for $i = 1 : H$
  { if $\omega_{j2} + t < \varepsilon$ and $\omega_{j2} + t > 0$
      $\omega_{j2} = \omega_{j2} + t$;
    elseif $\omega_{j2} - t < \varepsilon$ and $\omega_{j2} - t > 0$
      $\omega_{j2} = \omega_{j2} - t$;
    end
  }

$$r = \sum_{j2=1}^{H} w_{j2} \frac{1}{1+e^{-\lambda\mu_i}} - \sum_{j2=1}^{H} w_{j2} \frac{1}{1+e^{-\lambda\mu_j}};$$

}

3.2 The ULB-PSO-BPNN Algorithm

The detailed steps of ULB-PSO-BPNN are:

Step 1: Generate training samples and testing samples;

Step 2: Define the number of neurons in each layer and target error of NN; Define parameters of PSO: swarm size, particle dimension, inertia weight, acceleration coefficient and maximum number of iteration etc.

Step 3: Initialize the velocity and position of each particle randomly；

Step 4: Adjust the position of each particle according to the prior information；

Step 5: Train NN and compute fitness of particles according to training error;

Step 6: Compare the fitness and individual optimal extreme of each particle. If the former is better than the latter, it would be set as new individual optimal extreme;

Step 7: If the best individual optimal extreme is better than the current global optimal extreme, record the particle's number,adjust its position according to the prior information and then set it as new global optimal extreme；

Step 8: Update the velocity and position of each particle;

Step 9: Check that whether the stop condition of PSO is met. If it's true, stop search and record global optimal extreme and then go to step 11; else return step 6;

Step 10: Decode the global optimal extreme to the parameter of the BP neural network, including weights and thresholds;

Step 11:Continue to train neural network by BP algorithm until the target error is obtained;

Step 12: Compute the output of NN according to testing samples.

4. Experimental Results

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

$$y = (1-(40x/\pi)+2(40x/\pi)^2-0.4(40x/\pi)^3)e^{-20x/\pi}, \quad (5)$$

$$y = \sin(5x)/(5x). \quad (6)$$

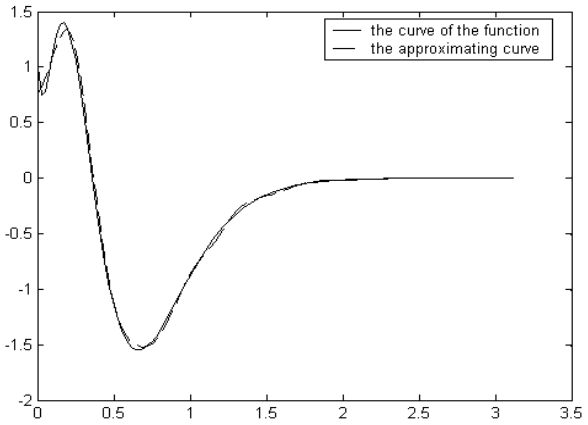


Figure 1. The Approximating Results of the ULB-PSO-BPNN Algorithm for Approximating Function: $y = (1-(40x/\pi)+2(40x/\pi)^2-0.4(40x/\pi)^3)e^{-20x/\pi}$

As for function (5), 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 $[0.0125,\pi-0.0125)$ at identically spaced intervals. Similarly, as for function (6), 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 ULB-PSO-BPNN algorithm for the above two functions are shown in Figures 1 and 2.

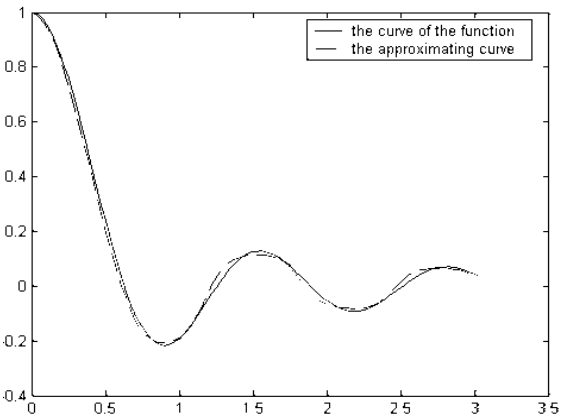


Figure 2. The Approximating Results of the ULB-PSO-BPNN Algorithm for Approximating Function: $y = \sin(5x) / (5x)$

The testing errors of the ULB-PSO-BPNN algorithm for the two functions are shown in Figures 3 and 4.

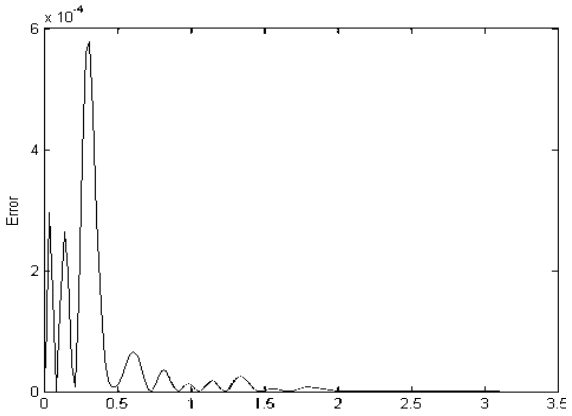


Figure 3. The Testing Errors of the ULB-PSO-BPNN Algorithm for Approximating Function: $y = (1-(40x/\pi)+2(40x/\pi)^2 - 0.4(40x/\pi)^3)e^{-20x/\pi}$

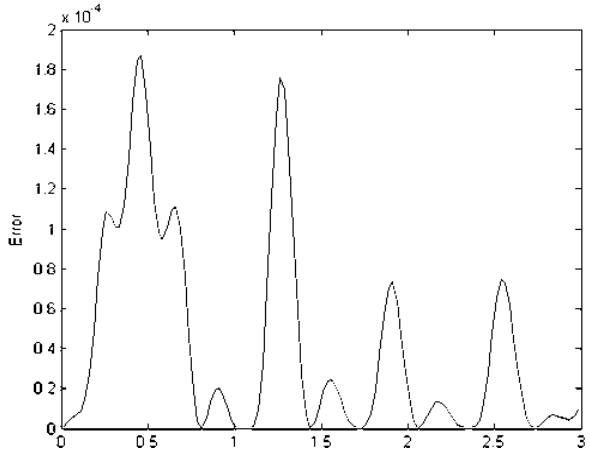


Figure 4. The Testing Errors of the ULB-PSO-BPNN Algorithm for Approximating Function: $y = \sin(5x) / (5x)$

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 Tables 1 and 2.

Table 1. The average values of MSE and Iteration Number for Approximating the Functio $y = (1-(40x/\pi)+2(40x/\pi)^2-0.4(40x/\pi)^3)e^{-20x/\pi}$ with Seven Learning Algorithms

Learning algorithms	Training MSE	Testing MSE	Iteration number
BP-FNN	5.3561e-4	4.6163e-4	30000
PSO-BPNN	3.0835e-4	2.2546e-4	15000
APSO-BPNN	2.6549e-4	1.6577e-4	15000
QPSO-BPNN	1.1752e-4	9.7861e-5	15000
APSOAEFDI-MHLA	4.6037e-5	2.3812e-5	15000
FOD-PSO-BPNN	4.5619e-5	2.3296e-5	15000
ULB-PSO-BPNN	4.8651e-5	2.7084e-5	15000

Table 2. The Average Values of MSE and Iteration Number for Approximating the Function $y = \sin(5x)/(5x)$ with Seven Learning Algorithms

Learning algorithms	Training MSE	Testing MSE	Iteration number
BPNN	1.5324e-4	1.4652e-4	30000
PSO-BPNN	8.2691e-5	7.9532e-5	15000
APSO-BPNN	7.6217e-5	7.5632e-5	15000
QPSO-BPNN	6.8073e-5	6.6594e-5	15000
APSOAEFDI-MHLA	2.0874e-5	1.8991e-5	15000
FOD -PSO-BPNN	2.1683e-5	1.9651e-5	15000
ULB-PSO-BPNN	1.9025e-5	1.8236e-5	15000

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

First, Figures 1 and 2 shows that the approximation curves are almost consistent with the curves of the functions. The new algorithm for function approximation works well.

Second, as for each function, the iteration numbers of BPNN are more than the other algorithms which use PSO or improved PSO. The results show that PSO can speed up the training process of NN because of its capability of global search.

Third, the testing errors of FOD -PSO-BPNN and ULB-PSO-BPNN are always less than the other five algorithms. The results demonstrate that PSO coupling with prior information can improve the approximating accuracy of NN effectively.

Finally, the testing error of FOD-PSO-BPNN is lowest in Table 1, but the testing error of ULB-PSO-BPNN is lowest in Table 2. It shows that the different prior information should be chosen when different functions are approximated.

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

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 number of PSO	Iteration number of BP	Testing MSE
PSO-BPNN	150	15000	2.2546e-4
QPSO-BPNN	150	15000	9.7861e-5
FOD-PSO-BPNN	100	10000	4.6872e-5
ULB-PSO-BPNN	100	10000	5.2841e-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 number of PSO	Iteration number of BP	Testing MSE
PSO-BPNN	150	15000	7.9532e-5
QPSO-BPNN	150	15000	6.6594e-5
FOD -PSO-BPNN	100	10000	3.6901 e-5
ULB -PSO-BPNN	100	10000	3.5023e-5

From Tables 3 and 4, we can find that the testing errors of FOD-PSO-BPNN and ULB-PSO-BPNN are still lowest when the iteration numbers of PSO and BP are all less than the other two algorithms. The results prove that the prior information helps the particle swarm converge to global optimum more quickly because the prior information guides the direction of particle flight, the search space of PSO is narrowed.

5. Conclusions

In this paper, an improved PSO algorithm is proposed. According to the prior information derived from the upper and lower bound of function, the corresponding particle positions are modified. Then the new PSO algorithm combined with BP is used to train NN 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.

Acknowledgement

This research was supported by the Natural Science Foundationfor Colleges and Universities of Jiangsu Province (No. 14KJD520002) and the Dr. Scientific Research Foundation of Jiangsu University of Science and Technology.

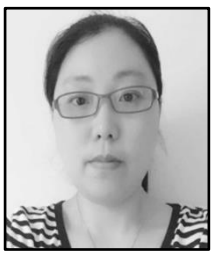
References

[1] J. Meng and Z. Sun, "Application of combined neural networks in nonlinear function approximation", Proceedings of the 3rd World Congress on Intelligent Control and Automation, **(2000)**.
[2] S. Lawrence, A. C. Tsoi and A. D. Back, "Function approximation with neural networks and local methods: bias, variance and smoothness", Proceedings of the Australian Conference on Neural Network, **(1996)**, Canberra, Australia.

[3] Y. Lin, W. Chang and J. Hsieh, "A particle swarm optimization approach to nonlinear rational filter Modeling", Expert Systems with Applications, vol. 34, no. 2, **(2008)**, p. 1194-1199.
[4] K. W. Chau, "Application of a PSO-based neural network in analysis of outcomes of construction claims", Automation in Construction, vol. 16, no. 5, **(2007)**, pp. 642-646.
[5] M. Lovbjerg and T. Krink, "Extending particle swarm optimizers with self-organized critically", Proceedings of the IEEE International Conference on Evolutionary Computation, **(2002)**, Honolulu.
[6] E. Ozcan and C. Mohan, "Particle swarm optimization: Surfing the waves", Proceedings of thecongress on Evolutionary Computation, **(1999)**, Piscataway, NJ.
[7] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer", Proceeding of IEEE International Conference on Computation Intelligence, **(1998)**.
[8] J. Sun, B. Feng and W. B. Xu, "Particle swarm optimization with particles having quantum behavior", Proceedings of the Congress on Evolutionary Computation, **(2004)**, Portland.
[9] T. M. Mitchell, "Machine Learning", McGraw-Hill, **(1997)**.
[10] P. Niyogi, F. Girosi and T. Poggio, "Incorporating Prior Information in Machine Learning by Creating Virtual Examples", Proceedings of the IEEE, **(1998)**.
[11] W. H. Joerding and J. L. Mcador, "Encoding a prior information in feed-forward networks", Neural Networks, no. 4, **(1991)**, pp. 847-856.
[12] F. Han and Q. Ling, "A new approach for function approximation incorporating adaptiveparticle swarm optimization and a priori information", Applied Mathematics and Computation, vol. 205, **(2008)**, pp. 792-798.
[13] J. Tu, Y. Zhan and F. Han, "An improved PSO algorithm coupling with priorinformation for function approximation", Information Technology Journal, vol. 10, no. 11, **(2011)**, pp. 2226-2231.
[14] J. Kennedy and R. C. Eberhart, "Particle swarm optimization", Proceedings of IEEE International Conference on Neural Networks, **(1995)** Piscataway, NJ.

Authors

JuanjuanTu, received the ME degree from Jiangsu University of Science and Technology, Zhenjiang, Jiangsu, in 2007 and the PhD degree from Jiangsu University, Zhenjiang, Jiangsu, in 2013. Her research interests include neural network, swarm intelligence optimization and bioinformatics.



Wenlan Zhou, received the ME degree from Jiangsu University of Science and Technology, Zhenjiang, Jiangsu, in 2007. Her research interests include pattern recognition and image processing.

Hongmei Li, received the ME degree from Jiangsu University of Science and Technology, Zhenjiang, Jiangsu, in 2014. Her major research area is intelligent computing.

