Optimized Neural Network Ensemble by Combination of Particle Swarm Optimization and Differential Evolution

Zeng-Shun Zhao[1,2,*], Xiang Feng[1], Fang Wei[1], Shi-Ku Wang[1], Mao-Yong Cao[1], and Zeng-Guang Hou[3]

[1] College of Information and Electrical Engineering, Shandong University of Science and Technology, Qingdao, 266590, China
[2] School of Control Science and Engineering, Shandong University, Jinan, 250061, China
[3] State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, P.R. China
zhaozengshun@gmail.com

Abstract. The Neural-Network Ensemble (NNE) is a very effective method where the outputs of separately trained neural networks are combined to perform the prediction. In this paper, we introduce the improved Neural Network Ensemble (INNE) in which each component forward neural network (FNN) is optimized by particle swarm optimization (PSO) and back-propagation (BP) algorithm. At the same time, the ensemble weights are trained by Particle Swarm Optimization and Differential Evolution cooperative algorithm(PSO-DE). We take two obviously different populations to construct our algorithm, in which one population is trained by PSO and the other is trained by DE. In addition, we incorporate the fitness value from last iteration into the velocity updating to enhance the global searching ability. Our experiments demonstrate that the improved NNE is superior to existing popular NNE.

Keywords: Neural Network Ensemble, Back-Propagation, Particle Swarm Optimization, Differential Evolution.

1 Introduction

Neural Network Ensemble (NNE) is a learning mechanism which has a collection of a finite number of neural networks trained for the same task. In Hansen and Salamon's work [1], it has been first proposed. Its main idea is that the predicting ability of a neural network system could be significantly improved by assembling a set of neural networks, for example, training many neural networks and then combining their predictions in some way [2]. But only by averaging, the combined prediction would not be effective, because in some cases, maybe some components of ensemble behave unsatisfactory. In [3], authors thought that it might be better to ensemble some components other than all of the trained neural networks; they introduced Genetic

[*] Corresponding author.

algorithm based selective ensembles (GASEN), which employed genetic algorithm to evolve the weights assigned to each FNN for the best appropriate prediction. In [4], Kennedy and Eberhart put forward the binary particle swarm optimization (BiPSO) to optimize the NNE: in the BiPSO, the weight of each FNN could be zero or 1, and the ensemble problem of NNE would be transformed into selecting the best appropriate FNN set by PSO. Another version of PSO, denoted as DePSO, in which the weight of each FNN could be decimal number.

The FNN often adopts the BP algorithm to optimize the weights. However, BP may lead to a failure in finding a global optimal solution [5]. But on the other part, the gradient descending method of BP could achieve higher convergent accuracy and faster convergent speed around the global optimum.

The PSO algorithm is showed to converge rapidly during the initial stages of a global search. But around global optimum, the search process may become very slow, the improvement decreasing gradually with the searching iterating. Another shortcoming is that the particles would easily oscillate in different sinusoidal waves, converging quickly, sometimes prematurely [6] [7]. In [8], the authors proposed PSO and BP couple-algorithm to train the weights of FNN, where the hybrid algorithm could make use of both global searching ability of the PSO and local searching ability of the BP algorithm. [9] proposed a new kind of hybrid method which was based on fuzzy set theory and used PSO-BP couple algorithm to determine the weight of different FNN, then synthesized their assessment result to form the final output according to the weight.

In our paper, we propose the improved PSO-BP-NNE mode, which means that we use PSO and BP to train each component FNN, and then use the Particle Swarm Optimization and Differential Evolution cooperative algorithm(PSO-DE) to optimize the NNE. There are two stages: In the FNN training stage, firstly, we use PSO to train each component FNN, when the constrain-condition is reached, we apply the BP algorithm into training until the new termination condition reached. In the NNE training stage, we present the multi-populations cooperative optimization (PSO and DE) to train the weight of each component FNN. In addition, we introduce an improved PSO algorithm which incorporates the fitness function into the velocity updating. In our experiment, the proposed algorithm is verified superior to the general NNE which is optimized by single algorithm.

2 Component Neural Network Optimized by PSO and BP

The gradient descending technique proposed by Werbos [10], is widely used in optimizing and training FNN. But it has its own disadvantage which is sensitive to the initial weight vector, often leading to a different result by virtue of different weight vector. The disadvantage leads trapping in a local solution which is bias to the best solution. But it could achieve faster convergent speed around global optimum, due to the reasons above, we introduce the PSO and BP couple-algorithm to optimize the FNN. The detailed gradient descending technique is described in [10] and [11].

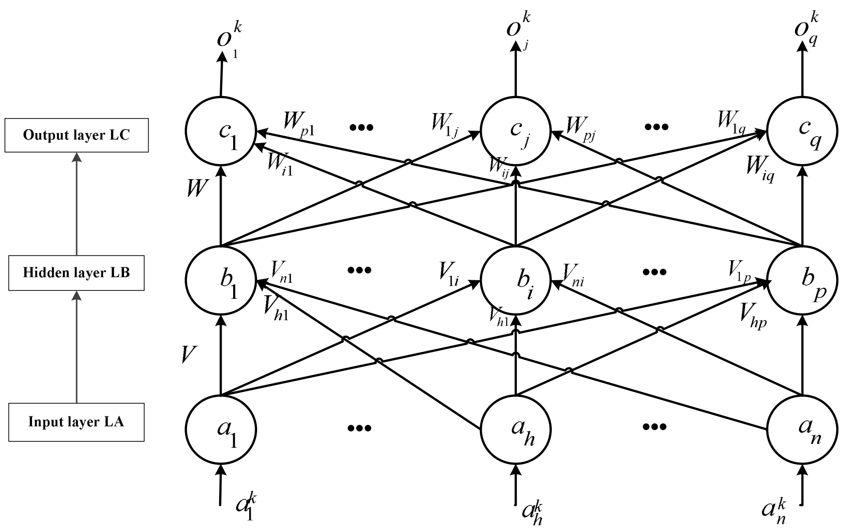

Fig. 1. The forward neural network architecture

The idea of BP is to make the error back-propagate to update the parameters of FNN, and the parameters include two sections: one is between the input-layer and hidden-layer, the other is between hidden-layer and output-layer. If we suppose that $X = \{x_1, x_2 ... x_m\}$ has m input samples and $Y = \{y_1, y_2 ... y_n\}$ has n output results. There are p neurons in hidden-layer and q neurons in output-layer. The thresholds of hidden neurons are $\theta^1 = \{\theta_1^1, \theta_2^1 ... \theta_p^1\}$ and thresholds of the output-layer are $\theta^2 = \{\theta_1^2, \theta_2^2 ... \theta_q^2\}$. We suppose the weights between input-layer and hidden-layer are $V = \{v_{11}, v_{12} .. v_{21} .. v_{np}\}$, weights between hidden-layer and output-layer are $W = \{w_{11}, w_{12} .. w_{21} .. w_{pq}\}$. The transition function of hidden-layer is $f(\cdot)$ and the transition function of output-layer is $g(\cdot)$.

We suppose the error between final-output and the expected output is E.

$$E = \frac{1}{2}\sum_{k=1}^{n}\{y_k - g[\sum_{j=1}^{p} w_{kj} f(\sum_{i=1}^{m} v_{ij} x_i - \theta_j^1) - \theta_k^2]\}^2 \tag{1}$$

We could get the updating formula of the two weights, as follows:

$$v_{ji}(t+1) = v_{ji}(t) + \Delta v_{ji} = v_{ji}(t) - \eta^1 \frac{\partial E}{\partial v_{ji}} \tag{2}$$

$$w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj} = w_{kj}(t) - \eta^2 \frac{\partial E}{\partial w_{kh}} \tag{3}$$

We could get the updating formula of the two thresholds, as follows:

$$\theta_j^1(t+1) = \theta_j^1(t) + \Delta\theta_j^1 = \theta_j^1(t) + \eta^1 \frac{\partial E}{\partial \theta_j^1} \tag{4}$$

$$\theta_k^2(t+1) = \theta_k^2(t) + \Delta\theta_k^2 = \theta_k^2(t) + \eta^2 \frac{\partial E}{\partial \theta_k^2} \tag{5}$$

Where η^1, η^2 is the learning rate.

But it should be noticed that the learning rate which controls the convergence to a local optimal solution is often determined by experiments or experience. If it is not ideal enough, it would easily result in oscillating of the network and could not converge rapidly.

The PSO algorithm could be described as a swarm of birds or pigeons hovering in the sky for food. We assume the pigeon swarm as a random particle swarm, and each particle stands for one bird. Every bird has its own location and flying-velocity. One swarm has m particles, and the number of dimensions of every particle is n, denoted as $X_i = (x_{i1}, x_{i2} ... x_{in})$ and $V_i = (v_{i1}, v_{i2} ... v_{in})$ where X_i and V_i are the position and velocity of the $i-th$ particle in n dimensional space. At each step of iterations, the particles update their positions and velocities according to the two best values: One is $P_i = (p_{i1}, p_{i2} ... p_{in})$, representing the previous best value of $i-th$ particle up to the current step. Another is $G_i = (g_1, g_2 ... g_n)$, representing the best value of all particles in the population. After obtaining the two best values, each particle updates its position and velocity according to the following equations:

$$v_{i,d}^{k+1} = w \cdot v_{i,d}^k + c_1 \cdot rand() \cdot (pbest_{i,d}^k - x_{i,d}^k) + c_2 \cdot rand() \cdot (gbest_d^k - x_{i,d}^k) \tag{6}$$

$$x_{i,d}^{k+1} = x_{i,d}^k + v_{i,d}^{k+1} \tag{7}$$

Here, rand() is a random number in the range [0, 1] generated according to a uniform probability distribution. In the general PSO, the learning actors c_1 and c_2 are positive constants, usually $c_1 = 2.8$, $c_2 = 1.3$. w^t is the inertial weight used to balance global and local searching. The detailed description could be referred in [12].

In the general PSO mechanism, the fitness values are used to select the best solutions. But the direct relation between the sequential iterations is usually ignored. In many cases, the values of the fitness mean the distance between the current location and the real best location. Incorporating the fitness value could enhance the global search-ability and diversity of the particle swarm.

Based on the motivation above, we introduce the improved PSO:

```
if   $fitness(x_i^k) > \theta$   ($\theta$ is the given threshold )
         $\hat{v}_i^k = fitness(x_i^k) \times v_i^k$
else     $\hat{v}_i^k = fitness(x_i^k) \times \xi \times v_i^k$   (Where $\xi$ is the acceleration constant)
endif
```

The improved updating equations list as follows:

$$v_{i,d}^{k+1} = w \cdot \hat{v}_i^k + c_1 \cdot rand() \cdot (pbest_{i,d}^k - x_{i,d}^k) + c_2 \cdot rand() \cdot (gbest_d^k - x_{i,d}^k) \tag{8}$$

$$x_{i,d}^{k+1} = x_{i,d}^k + v_{i,d}^{k+1} \tag{9}$$

The procedures for PSO–BP couple algorithm could be summarized as follows:

Step 1: Initialize the swarm of PSO: get M particles, set the initial weight w and learning factor c_1, c_2, the maximal iterative generations $T_{\max-pso}$ and $T_{\max-BP}$.

Step 2: Evaluate the fitness of each particle of PSO, $pbest_i$ represents the previous best value of the $i-th$ particle up to the current step. $gbest$ represents the best value of all the particles in the population.

Step 3: Do step 3 until $t > T_{\max-pso}$ or do step4 if the best position has not changed for several iterations, or else, return to the step 4.

Step 4: Do BP algorithm until $t > T_{\max-BP}$.

```
Step 5: if  $fitness(gbest) < E$  (E is the given threshold)
            Output the prediction and MSE
        else  Continue to do BP for several iterations.
            Output the prediction and MSE
        endif
```

3 Neural Network Ensemble Optimized by DE-PSO

The authors have described the NNE in detail in[13][14]. Having obtained each refined component FNN as described in Section II, we would concentrate on how to combine the output of each component FNN.

$$\bar{f}(x) = \sum_{i=1}^{n} \overline{w}_i f_i(x) \tag{10}$$

$$\overline{w}_i = \frac{w_i}{\sum_{i=1}^{n} w_i}, 0 < w_i < 1 \tag{11}$$

where $f_i(x)$ represents the output of the $i-th$ FNN and $\overline{w}_i$ represents the importance of the $i-th$ FNN. Our idea is that, for the best appropriate prediction how to optimize $\overline{w}_i$ of each sub-network, which corresponds to the solution of the optimization problem of the particles. But in [13], the authors recommended to average the weight of each sub-network. In this paper, we introduce the multi-population cooperative algorithm which could not only avoid trapping into the local solution, but also increase the diversity of particles. Here we introduce another global-searching algorithm, differential evolution algorithm [15]. DE is also a floating-point encoded evolutionary algorithm for global optimization over continuous spaces, but it creates new candidate solutions by combining the parent individual and several other individuals of the same population. It consists of selection, crossover and mutation.

In [16] [17], authors utilized DE to optimize PSO to improve the efficiency and precision. The cooperative algorithm tends to compensate for disadvantage of the individual method and could be apt to the best solution. We also incorporate this idea, but in every iteration, the two populations optimized respectively by different algorithms would be compared and to select the best appropriate solution which determines the evolution direction.

Our architecture is as follows.

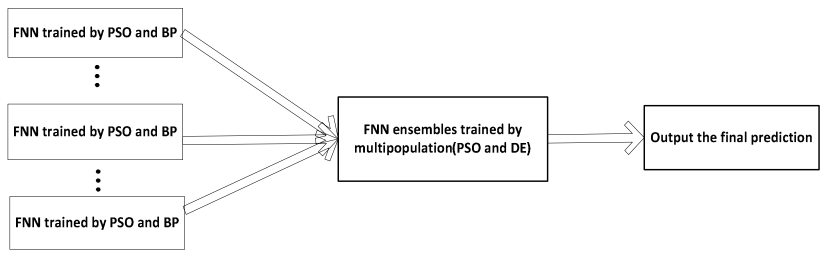
FNN trained by PSO and BP
FNN trained by PSO and BP
FNN trained by PSO and BP
FNN ensembles trained by multipopulation(PSO and DE)
Output the final prediction

Fig. 2. The sketch diagram of the whole mechanism

The procedure for NNE-multi-population algorithm could be summarized as follows:

Step 1: Initialize the weight of each FNN which has been optimized by PSO and BP.

Step 2: Each particle represents a set of weights which means that each dimension represents one weight of each component FNN .The population is duplicated into 2 identical swarms.

Step 3: One swarm is optimized by PSO and the other is optimized by DE respectively.

Step 4: After each Step3, the $gbest\_pso$ and $gbest\_DE$ are calculated.

$$gbest = \max(gbest\_pso, gbest\_DE)$$

Step 5: Do the Step 3-Step 4 loop until the Max-iteration is reached.

Step 6: Output the MSE.

4 Experiment

To test the efficiency of the improved NNE, we perform the comprehensive experiments to compare different optimization methods. We select the input-sample set for training from $X = \{-4:0.08:4\}$ with 100 samples, we could get the expected output via the equation $y = 1.1\times(1-x+2\cdot x^2)\cdot e^{(-x^2/2)}$. We suppose the test-sample $\hat{X} = \{-3.96:0.08:3.96\}$ with 100 samples. We regard the MSE, the mean square error between the real-output and the expected-output, as the measure variable.

The performance is compared between various ensemble ways with the different component FNN and different ways to combine the output of each component FNN. In our experiment, there are three kinds of component FNN: optimized by BP, optimized by PSO, optimized by PSO and BP. The ensemble weights of NNE are optimized in five ways: simple averaging, general PSO, improved PSO, multi-population improved PSO and multi-population improved PSO and DE, which are listed in the following table.

Table 1. The train-MSE and test-MSE comparison between five ensemble ways with each FNN optimized by three ways

The optimized method of NNE	Each component FNN optimized by BP		Each component FNN optimized by PSO		Each component FNN optimized by PSO and BP	
	MSE-train	MSE-test	MSE-train	MSE-test	MSE-train	MSE-test
Simple average	0.4543	0.4154	0.0133	0.0134	5.1015e-007	4.9619e-007
General PSO	0.4324	0.4031	0.0056	0.0055	1.8851e-007	1.8320e-007
Improved PSO	0.3348	0.3095	0.0058	0.0057	1.4201e-007	1.4180e-007
Multi-population Improved PSO	0.2883	0.2616	0.0046	0.0043	7.0418e-008	6.7776e-008
Multi-population Improve PSO and DE	0.1997	0.1905	0.0041	0.0044	4.5633e-008	4.3873e-008

From Table I, the results which are related to individual networks optimized by different algorithm have been listed. It could see that, the individual network optimized by BP and PSO couple-algorithms does better than other algorithms. Among different NNE training algorithms, we could discover that the multi-population cooperative algorithm is superior to the other NNE trained algorithms.

5 Conclusion

In this paper, the superiority of individual networks optimized by different algorithm is analyzed, which reveals that in some cases the ensemble mechanism is superior to the simplex neural network. The weights of NNE also reveals the importance of individual networks, Experimental results show that multi-population cooperative algorithm is a promising ensemble approach that is superior to both averaging all and our other enumerating algorithms.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 60805028, 61175076), Natural Science Foundation of Shandong Province (ZR2010FM027), China Postdoctoral Science Foundation (2012M521336), Open Research Project under Grant 20120105 from SKLMCCS, SDUST Research Fund (2010KYTD101).

References

1. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Trans. Pattern Analysis and Machine Intelligence 12(10), 993–1001 (1990)
2. Perrone, M.P., Cooper, L.N.: When networks disagree: Ensemble methods for hybrid neural networks. Neural Networks for Speech and Image Processing. pp. 126–142 (1993)
3. Zhou, Z.-H., Wu, J., Tang, W.: Ensembling Neural Networks: Many Could Be Better Than All. Artificial Intelligence 137(1-2), 239–263 (2002)
4. Kennedy, J., Eberhart, R.: A discrete binary version of the particle swarm optimization. In: Proceedings IEEE International Conference on Computational Cybernatics and Simulation, Piscataway, pp. 4104–4108 (1997)
5. Gori, M., Tesi, A.: On the problem of local minima in back-propagation. IEEE Trans. Pattern Anal. Mach. Intell. 14(1), 76–86 (1992)
6. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intell. 1, 33–57 (2007), doi:10.1007/s11721-007-0002-0(2007)
7. Zhao, Z., Wang, J., Tian, Q., Cao, M.: Particle Swarm-Differential Evolution Cooperative Optimized Particle Filter. In: ICICIP, pp. 485–490 (2010)
8. Zhang, J., Zhang, J., Lok, T., Lyu, M.R.: A hybird particle swarm optimization-back-propagation algorithm for feedward neural netwrok train. Applied Mathematics and Computation 185, 1026–1037 (2007)
9. Yuan, H., Zhi, J., Liu, J.: Application of particle swarm optimization algorithm-based fuzzy BP neural network for target damage assessment. Scientific Research and Essays 6(15), 3109–3121 (2011)
10. Werbos, P.J.: Beyond regression: New tools for predictions and analysis in the behavioral science. Ph.D. Thesis, Harvard University (1974)
11. Vogl, T.P., Mangis, J.K., Rigler, A.K., Zink, W.T., Alkon, D.L.: Accelerating the convergence of the back-propagation method. Biological Cybernetics 59, 257–263 (1988)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. IEEE Conf. Neural Networks, Piscataway, pp. 1942–1948 (1995)
13. Optiz, D., Shavlik, J.: Actively searching for an effectively neural network ensemble. Connection Science 8(34), 337–353 (1996)
14. Valentini, G., Masulli, F.: Ensembles of Learning Machines. In: Marinaro, M., Tagliaferri, R. (eds.) WIRN VIETRI 2002. LNCS, vol. 2486, pp. 3–20. Springer, Heidelberg (2002)
15. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimization 11(4), 341–359 (1997)
16. Das, S., Abraham, A., Konar, A.: Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives. In: Liu, Y., Sun, A., Loh, H.T., Lu, W.F., Lim, E.-P. (eds.) Advances of Computational Intelligence in Industrial Systems. SCI, vol. 116, pp. 1–38. Springer, Heidelberg (2008)
17. Luitel, B., Venayagamoorthy, G.K.: Differential Evolution Particle Swarm Optimization for Digital Filter Design. In: Proc. 2008 IEEE Congress on Evolutionary Computation (CEC 2008), Crystal city, Washington, DC, USA, pp. 3954–3961 (July 2008)