# Training RBF Neural Network with Hybrid Particle Swarm Optimization

Haichang Gao[1], Boqin Feng[1], Yun Hou[1], and Li Zhu[2]

[1] School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China
gaohaich@gmail.com
[2] School of Software Engineering, Xi'an Jiaotong University, Xi'an 710049, China

**Abstract.** The particle swarm optimization (PSO) has been used to train neural networks. But the particles collapse so quickly that it exits a potentially dangerous stagnation characteristic, which would make it impossible to arrive at the global optimum. In this paper, a hybrid PSO with simulated annealing and Chaos search technique (HPSO) is adopted to solve this problem. The HPSO is proposed to train radial basis function (RBF) neural network. Benchmark function optimization and dataset classification problems (Iris, Glass, Wine and New-thyroid) experimental results demonstrate the effectiveness and efficiency of the proposed algorithm.

## 1 Introduction

Particle swarm optimization (PSO) is a new evolutionary computation technique introduced by Kennedy [1], which was inspired by social behaviors of birds. Similar to genetic algorithm (GA), PSO is a population based optimization tool [2]. But unlike GA, PSO has no evolution operators such as crossover and mutation. Compared with GA, PSO has some attractive advantages. It has memory, so knowledge of good solutions is retained by all particles. It has constructive cooperation between particles, particles in the swarm share information between them. PSO has been successfully applied in many areas: function optimization, artificial neural network training, fuzzy system control, and other areas [3]. But the particles collapse so quickly that it exits a potentially dangerous stagnation characteristic, which would make it impossible to arrive at the global optimum. In this paper, a hybrid PSO with simulated annealing and Chaos search technique (HPSO) is adopted to solve this problem.

## 2 RBF Neural Network

Radial basis function (RBF) networks were introduced into the neural network by Broomhead [4]. Due to the better approximation capabilities, simpler network structures and faster learning algorithms, RBF networks have been widely used in many fields.

A RBF neural network has a three-layer architecture with on feedback [5]. The input layer which consists of a set of source nodes connects the network to the environment. The hidden layer consists of H hidden neurons (radial basis units), with radial activation functions. Gaussian function is often selected as the activation function. The output of i-th hidden neuron, $z_i$, is a radial basis function that defines a spherical receptive field given by the following equation:

$$z_i = \Phi(\| x - c_i \|) = \exp(-\| x - c_i \|^2 /(2\sigma_i^2)) \text{ , } \forall i \text{ .} \tag{1}$$

where $c_i$ and $\sigma_i$ are the center and the width of the $i$-th hidden unit, respectively. Each neuron in the hidden layer has a substantial finite spherical activation region, determined by the Euclidean distance between input vector, $x$, and the center, $c_i$, of the function $z_i$ normalized with respect to the scaling factor $\sigma_i$. The output layer, a set of summation units, supplies the response of the network.

## 3 Hybird PSO Training RBF Neural Network

PSO has a strong ability finding the most optimistic result. But it has a disadvantage of local optimum. SA has a strong ability finding the global optimistic result, and it can avoid the problem of local optimum. Chaos movement can go through all states unrepeated according to the rule of itself in some area. So, combining PSO with Chaos and SA, learning from other's strong point and offset one's weak point each other, the hybrid PSO strategy (HPSO) is proposed.

### 3.1 Particle Swarm Optimization

The basic PSO model consists of a swarm of particles moving in an $n$-dimensional search space. Each particle has a position represented by a position-vector $X$ and a velocity represented by a velocity-vector $V$. Particles move to trying to find the solution for the problem being solved. They find the global best solution by simply adjusting the trajectory of each individual towards its own best location and towards the best particle of the swarm at each time step.

The position and the velocity of the $i$-th particle in the $n$-dimensional search space can be represented as $X_i = [x_{i1}, x_{i2}, \cdots, x_{in}]$ and $V_i = [v_{i1}, v_{i2}, \cdots, v_{in}]$, respectively. Each particle has its own best position $P_{id}$, corresponding to the personal best objective value obtained so far. The global best particle is denoted by $P_{gd}$, which represents the best particle found so far. At each iteration step, the velocity is updated and the particle is moved to a new position. The update of the velocity from the previous velocity to the new velocity is calculated as follows:

$$V_{id}' = \omega \cdot V_{id} + G_1 \cdot rand() \cdot (P_{id} - X_{id}) + G_2 \cdot rand() \cdot (P_{gd} - X_{id}) \text{ .} \tag{2}$$

where $G_1$ and $G_2$ are constants called acceleration coefficients, $\omega$ is called the inertia factor, $rand()$ is random number uniformly distributed in the range of [0,1].

The new position is determined by the sum of the previous position and the new velocity, and it can be calculated according to the following equation:

$$X_{id}' = X_{id} + V_{id} . \tag{3}$$

Due to the simple concept, easy implementation and quick convergence, nowadays PSO has gained much attention and wide application. But the performance of simple PSO greatly depends on its parameters, and it often suffers the problem of being trapped in local optima. Researchers have analyzed it empirically [6] and theoretically [7], which have shown that the particles oscillate in different sinusoidal waves and converging quickly, sometimes prematurely, especially for PSO with small inertia factor $\omega$ or acceleration coefficients $G_1$ and $G_2$ .

## 3.2 Chaos Optimization

Chaos movement can go through all states unrepeated according to the rule of itself in some area. Chaos has three important dynamic properties: the sensitive dependence on initial conditions, the intrinsic stochastic property and ergodicity. Chaos is in essence deeply related with evolution. In chaos theory, biologic evolution is regarded as feedback randomness, while this randomness is not caused by outside disturbance but intrinsic element [8].

Logistic equation [9] is brought forward for description of the evolution of biologic populations. It is the most common and simple chaotic function:

$$x_{n+1} = L \cdot x_n (1 - x_n) . \tag{4}$$

where, $L$ is a control parameter which is between 0 and 4.0. When $L$=4.0, the system is proved to be in chaotic state. Given arbitrary initial value that is in (0,1) but not equal with 0.25, 0.5 and 0.75, chaos trajectory will finally search non-repeatedly any point in (0,1).

If the target function of continuous object problem that to be optimized is:

$$f^3 = f(x_i^3) = \min f(x_i) ,\ x_i \in [a_i, b_i],\ \text{i=1,2,...,n.} \tag{5}$$

Then the process of the chaos optimization strategy can be described as follows:

*Step 1:* algorithm initialization. Let $k = 1$ , $k' = 1$ , $x_i^k = x_i(0)$ , $x_i^3 = x_i(0)$ , $f^3 = f(0)$ , $a_i^{k'} = a_i$ , $b_i^{k'} = b_i$ . Where, $k$ is the iterative symbol of chaos parameters. $k'$ is the refine search symbol. $x_i^3$ is the best chaos variable found currently. $f^3$ is the current best solution that initialized as a biggish number.
*Step 2:* map the chaos variable $x_i^k$ to the optimization variable area, get $mx_i^k$ :

$$mx_i^k = a_i^{k'} + x_i^k (b_i^{k'} - a_i^{k'}) . \tag{6}$$

*Step 3:* search according to the chaos optimization strategy. $f^3 = f(mx_i^k)$ , $x_i^3 = x_i^k$ , if $f(mx_i^k) < f^3$ . Otherwise, go on.

*Step 4:* let $k = k + 1$ , $x_i^k = 4x_i^k (1 - x_i^k)$ , repeat step 2 and 3 until $f^3$ keep unchanged in certain steps.
*Step 5:* reduce the search scale of chaos variable:

$$a_i^{k'+1} = mx_i^3 - C(b_i^{k'} - a_i^{k'}) \text{ , } b_i^{k'+1} = mx_i^3 + C(b_i^{k'} - a_i^{k'}) \text{ .} \tag{7}$$

where, adjustment coefficient $C \in (0,0.5)$ , $mx_i^3$ is the best solution currently.

*Step 6:* revert optimization variable $x_i^3$ :

$$x_i^3 = (mx_i^3 - a_i^{k'+1}) / (b_i^{k'+1} - a_i^{k'+1}) \text{ .} \tag{8}$$

Repeat step 2 to 5 using new chaos variable $y_i^k = (1-A)x_i^3 + Ax_i^k$ , where A is a small number. Let $k' = k' + 1$, until $f^3$ keep unchanged in certain steps.

*Step 7:* finish the calculate process after several repeating of step 5 and 6. The final $mx_i^3$ is the best optimization variable, and $f^3$ is the best solution.

## 3.3 Simulated Annealing

SA is based on the idea of neighborhood search. Kirkpatrick [10] suggested a form of SA could be used to solve complex optimization problems. The algorithm works by selecting candidate solutions which are in the neighborhood of the given candidate solution. SA attempts to avoid entrapment in a local optimum by sometimes accepting a move that deteriorates the value of the objective function. With the help of the distribution scheme, SA can provide a reasonable control over the initial temperature and cooling schedule so that it performs effective exploration and good confidence in the solution quality.

In Annealing function construction, exponential cooling schedule is used to adjust the temperature $t_{k+1} = \mu \cdot t_k$ , where $\mu \in (0,1)$ is a decrease rate. It is often believed to be a good cooling method, because it provides a rather good compromise between a computationally fast schedule and the ability to reach low-energy state.

## 3.4 Training Algorithm of HPSO

HPSO algorithm training RBF neural network can be summarized as follows:

*Step 1:* Initialize the structure, activation function and objective function of HPSO.

*Step 2:* Initialize the algorithm parameters of HPSO (i.e. initialize velocities $X_i$ and positions $V_i$ randomly. Initialize temperature $T_0$ and cooling parameter $\alpha$ . Initialize $P_{id}$ , equal to $P_{gd}$ , with the index of the particle with the best position). Set a limit to particles' velocities and positions.

*Step 3:* Evaluate and store initial position and fitness of each particle. Evaluate and store the global best position and fitness of the swarm.

*Step 4:* Update particles' velocities $V_i$ and positions $X_i$ by equation (2) and (3). Update the individual best position and fitness of each particle.

*Step 5:* Implement the Chaos search for the best particle. Decrease the search space according to equation (7) and (8). Update the global best position and fitness of the swarm.

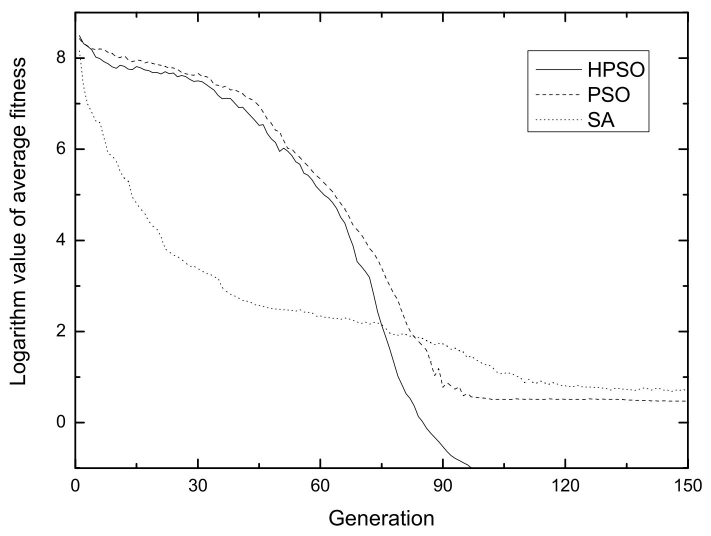

**Fig. 1.** Average fitness logarithm value curve of function *F*

*Step 6:* Perform annealing operation, Decrease temperature $t_{k+1} = update(t_k)$ and set k=k+1.

*Step 7:* If the stopping criterion is not satisfied, go to step 4. Otherwise, output the best solution found so far.

## 4 Experiments

HPSO was applied to benchmark function optimization and dataset classification problems (Iris, Glass, Wine and New-thyroid) experiments in this section. Benchmark function optimization experiment was carried to demonstrate the effectiveness of the proposed algorithm in detail.

### 4.1 Benchmark Function Optimization

Three algorithms (HPSO, simple PSO [11] and SA) were compared on DeJong benchmark function optimization. The DeJong function is following:

$$F = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 . \tag{9}$$

where $-2.048 \le x_i \le 2.048$ , (i=1,2). The function is continuous and multimodal; x*=1, with $f(1,1) = 0$.

From the average fitness logarithm value cure of function F in figure 1, we can see HPSO performs PSO and SA in finding the global optimistic result, and it can avoid the problem of local optimum effectively.

### 4.2 Benchmark Datasets Experiment

Four datasets, which are all classification problems and can be got from UCI dataset house [12], was selected to carry the experiment. The attribute, class and instance of each dataset can be found in table 1. Each method runs 10 times on every dataset, and the average value was selected as the experiment result.

**Table 1.** Datasets characteristic used for experiment

| Dataset | Example number | Input attribution | Output attribution |
|---|---|---|---|
| Iris | 150 | 4 | 3 |
| Glass | 214 | 9 | 7 |
| Wine | 178 | 13 | 3 |
| New-thyroid | 215 | 5 | 3 |

Three training algorithm (HPSO, PSO, and newrb) were compared [11]. The newrb routine was included in Matlab neural network toolbox as standard training algorithm for RBF neural network. The parameters of PSO and HPSO were set as follows: weight $\omega$ decreasing linearly between 0.8 and 0.2, acceleration coefficients $G_1 = G_2 = 2.5$. The initiation temperature of HPSO is 1000. The test results have been listed on table 2, which are Error rate of three methods on different dataset.

**Table 2.** Comparative accuracy rate of three algorithms on different datasets

| Dataset | HPSO | | PSO | | newrb | |
|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test |
| Iris | 0.9989 | 0.9875 | 0.99 | 0.98 | 0.9850 | 0.9560 |
| Glass | 0.9124 | 0.7572 | 0.8042 | 0.6620 | 0.9850 | 0.6174 |
| Wine | 0.9991 | 0.9668 | 1 | 0.9631 | 0.9375 | 0.6554 |
| New-thyroid | 0.9763 | 0.9534 | 0.9650 | 0.9444 | 0.9240 | 0.6204 |

From table 2, it can be seen that the accurate rate of train set and test set outperform those of simple PSO and newrb. So, the HPSO algorithm proposed for RBF neural network in this paper is more effective.

## 5 Conclusion

This paper presents a hybrid PSO with simulated annealing and Chaos search technique to train RBF neural network. The HPSO algorithm combined the strong ability of PSO, SA, and Chaos. They can learn from other's strong point and offset one's weak point each other. Benchmark function optimization and dataset classification problems (Iris, Glass, Wine and New-thyroid) experimental results demonstrate the effectiveness and efficiency of the proposed algorithm.

## Acknowledgements

The authors would like to thank the anonymous reviewers for their careful reading of this paper and for their helpful comments. This work was supported by the National High Technology Development Plan of China (863) under grant no. 2003AA1Z2610.

## References

1. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, Perth, Australia (1995) 1942–1948
2. Eberhart, R.C., Shi, Y.: Comparison between Genetic Algorithm and Particle Swarm Optimization. In. Proceedings of 7th Annual Conference on Evolutionary Computation (1998) 611–616
3. Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers, Inc., San Francisco, CA (2001)
4. Broomhead, D, Lowe, D.: Multivariable Functional Interpolation and Adaptive Networks. Complex Systems 2 (1998) 321–355
5. Catelani, M., Fort, A.: Fault Diagnosis of Electronic Analog Circuits Using a Radial Basis Function Network Classifier. Measurement 28(2000)147–158
6. Kennedy, J.: Bare Bones Particle Swarms. In: Proceedings of IEEE Swarm Intelligence Symposium, (2003)80–87
7. Cristian, T.I.: The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection. Information Processing Letters 85(6) (2003)317–325
8. Zhang, T., Wang, H.W., Wang Z.C.: Mutative Scale Chaos Optimization Algorithm and Its Application. Control and Decision 14(3) (1999)285–288
9. Moon Francis C. Chaotic and Fractal Dynamics, an Introduction for Applied Scientists and Engineers. New York: John Wiley & Sons, (1992)
10. Kirkpatrick, S., Gelat, J.C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 4596(220) (1983)671–680
11. Liu, Y., Qing, Z., Shi, Z.W.: Training Radial Basis Function Network with Particle swarms, ISNN04, Springer-Verlag Berlin Heidelberg (2004)317–322
12. Blake, C., Keogh, E., Merz, C.J.: UCI Repository of Machine Learning Databases, www.ic.uci.edu/~mlearn/MLRepository.htm (2003)