CIMSA 2009 - International Conference on Computational Intelligence for Measurement Systems and Applications
Hong Kong, China
May 11-13, 2009

Adjusting the Parameters of Radial Basis Function Networks Using Particle Swarm Optimization

A. Esmaeili
School of Computer Engineering
Iran University of Science and Technology
Tehran, Iran
esmaeili@comp.iust.ac.ir

N. Mozayani
School of Computer Engineering
Iran University of Science and Technology
Tehran, Iran
mozayani@iust.ac.ir

Abstract— **Particle Swarm Optimization (PSO), a new promising evolutionary optimization technique, has a wide range of application in optimization problems including training of artificial neural networks. In this paper, an attempt is made to completely train a RBF neural network architecture including the centers, optimum spreads, and the number of hidden units. The proposed method has been evaluated on some benchmark problems: Iris, Wine, Glass, New-thyroid and its accuracy was compared with other algorithms. The results show its strong generalization ability.**

Keywords- RBF Networks; PSO; Neural Networks; Neural Networks Training

I. INTRODUCTION

Radial basis function (RBF) networks were introduced into neural network by Broomhead and Lowe [2]. Being universal function approximators, these networks are asymptotically Bayes-optimal classifiers [7], and in addition, are considered to be an improved alternative of the classical multilayer neural networks in generalization ability, computational efficiency and biological plausibility.

The construction of a radial basis function (RBF) network, in its most basic form, involves three layers with entirely different roles [5]: the input layer which is made up of source nodes, the second and the only hidden layer which applies a nonlinear transformation from the input space to the hidden space, mostly of high dimensionality, by means of radial functions, and finally an output layer which performs a linear transformation (Fig.1).

This kind of network, in comparison with the other nonparametric classifiers, has many tunable parameters:

- The type of radial function to be used in the hidden units.
- The distance type.
- The center of the radial functions (location of the hidden units).
- The spread or radius of the radial functions.

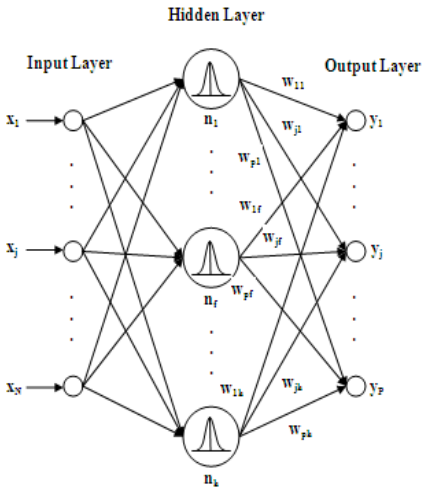

Figure 1. Architecture of an RBF network

As for the hidden units, Gaussian function is often used as the radial function and Euclidean distance as the distance type. In this case, the output of the i-th hidden unit with center μ_i and spread σ_i is given as follows:

$$\phi_i(\mathrm{x}) = \phi\left(\|\mathrm{x}-\mu_i\|;\sigma_i\right) = e^{\left(-\frac{\|\mathrm{x}-\mu_i\|^2}{2\sigma_i^2}\right)}, \forall i \quad (1)$$

Training an RBF network consists of finding the values for these parameters, such that the overall approximation or

classification error is reduced. The values chosen for the centers and the spread of the radial functions have a great effect on the generalization abilities of the network. Many algorithms have been proposed for finding the centers of an RBF network in literature such as: random subset selection, various clustering algorithms such as K-means [3] and vector quantization [4], sequential growing, and most recently a training algorithm in which the locations and the number of hidden units are tried to be optimized using particle swarms [6]. In all of these works the spread of the radial functions of the hidden units are set either to a fixed value or a value obtained using heuristic methods.

In this work an attempt is made to optimize the centers and corresponding spreads of the hidden units using particle swarm optimization. The experimental results on some benchmark problems show its strong generalization ability.

II. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO), introduced by Kennedy [1], mimics the behavior of a swarm of insects or a school of fish. If one of the particles discovers a good path to food the rest of the swarm will be able to follow instantly even if they are far away in the swarm. Swarm behavior is modeled by particles in multidimensional space that have two characteristics: a position and a velocity. These particles wander around the hyperspace and remember the best position that they have discovered. They communicate good positions to each other and adjust their own position and velocity based on these good positions. If we would like to describe the process used in this algorithm more technically we would follow these steps: the swarm of particles at first is initialized with a population of random candidate solutions. They move iteratively through the d-dimension problem space to search the new solutions, where the fitness, f, can be calculated as the certain qualities measure. Each particle has a position represented by a position-vector xi (i is the index of the particle), and a velocity represented by a velocity-vector vi. Each particle remembers its own best position so far in a vector Pi =(pi1, pi2, …, pid), and its j-th dimensional value is pij . The best position-vector among the swarm so far (global best) is then stored in a vector Pg, and its j-th dimensional value is pgj. During the iteration time t, the update of the velocity from the previous velocity to the new velocity is determined by Eq. (2). The new position is then determined by the sum of the previous position and the new velocity by Eq. (3).

$$v_{ij}(t+1) = w(t)v_{ij}(t) + c_1 r_1(p_{ij}(t) - x_{ij}(t)) + c_2 r_2(p_{gj}(t) - x_{ij}(t)) \quad (2)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (3)$$

where *w(t)* is called as the inertia factor, r_1 and r_2 are the random numbers, which are used to maintain the diversity of the population, and are uniformly distributed in the interval [0,1] for the *j*-th dimension of the *i*-th particle. c_1 is a positive constant, called as coefficient of the self-recognition component, c_2 is a positive constant, called as coefficient of the social component. The following pseudo-code illustrates the algorithm.

Algorithm.1. Particle Swarm Optimization Algorithm

```
01. Initialize the size of the particle swarm n, and other parameters.
02. Initialize the positions and the velocities for all the particles randomly.
03. While (the end criterion is not met) do
04.     t = t + 1;
05.     Calculate the fitness value of each particle;
06.     P_g = argmin_n^{i=1} (f(P_g(t − 1)), f(x_1(t)), f(x_2(t)), · · · , f(x_i(t)), · · · , f(x_n(t)));
07.     For i= 1 to n
08.         P_i (t) = argmin_n^{i=1}(f(P_i (t − 1)), f(x_i(t));
09.         For j = 1 to Dimension
10.             Update the j-th dimension value of x_i and v_i
10.                 according to Eqs.(2), (3);
12.         Next j
13.     Next i
14. End
```

III. CONFIGURATION OF THE PSO TRAINING ALGORITHM

As stated before, the purpose of training in an RBF network is to determine the centers, numbers, and spread of the hidden units and the weights of connections from hidden units to the output layer. We would like to determine proper values for these training parameters by means of PSO. But encoding all of these parameters into a particle makes the length of the particle too long and hence the searching space becomes too large. Therefore we try to encode the centers and the spreads of the hidden units in a particle (Fig. 2), and compute the weights of the output connections by analytic methods. If the maximum number of the hidden units is given by hmax , the proposed encoding of the network architecture into a particle is as follows:

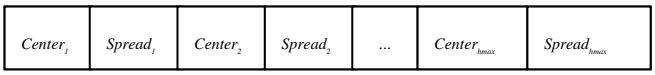

Figure 2. Network encoding in a particle

where $Center_i$ contains the center of the radial function used in *i*-th hidden unit and its dimension equals the dimension of input space. $Spread_i$ contains the width used in *i*-th hidden unit and is a real number. If $Spread_i < 0$, then the *i*-th hidden unit will not be considered in the architecture of the network. By this representation we can tune three training parameters using PSO with a reasonable particle length. Once these values are determined through a stochastic search, the weights of the connections from hidden units to the output layer can be easily determined using SVD analytic method.

PSO like other evolutionary algorithms needs a fitness function to be guided to a desired point. For this proposed method the fitness function will be designed considering the network accuracy and complexity. For the accuracy the RBF network in this article the mean squared network (MSE) is used. For this purpose the network with the architecture obtained from the particles in the swarm and the weights

computed analytically is fed with the training dataset, the corresponding outputs are collected and compared with the desired outputs. The complexity of the network can be tracked by considering the number of the hidden units. The lower the number of hidden units is, the more desirable the fitness will be. Therefore the fitness function would be as follows:

$$fitness = \frac{1}{N}\sum_{i=1}^{N}\|t_i - o_i\|^2 + k\frac{SizeHidden}{MaxHidden}. \quad (4)$$

where N is the number of training patterns, k is a balancing factor, t_i, o_i are the desired output and actual network output respectively.

IV. EXPERIMENTS

The proposed method has been tested on four datasets: Iris, Glass, Wine, and New-thyroid, all from the UCI repository of machine learning [6].The characteristic of these datasets has been brought in table 1. The results obtained are compared with two other methods, the *newrb* training method implemented in Matlab neural network toolbox as an standard RBF training algorithm and the simple-PSO [6]. The parameters of the PSO algorithm in the proposed method have been chosen as follows: learning rates, c_1 and c_2, were both set to 2, the inertia weigh, $w(t)$, was set to vary linearly from 0.9 to 0.3 during the run of algorithm, V_{max} was set to 5, the maximum number of hidden units was set to 20, and maximum number of iterations was set to 5000.

TABLE 1. CHARACHTERISTICS OF THE DATASETS

Dataset	Number of instances	Number of features	Number of classes
Iris	150	4	3
Wine	178	13	3
Glass	214	9	7
New-thyroid	215	5	3

All of the training algorithms mentioned above were conducted 20 runs. In each run the data set was randomly partitioned into two parts: training set and test set. The accuracy of the algorithms has been measured on both of these parts and the average accuracy during different runs is listed as in table 2 (CmPSO refers to the proposed method in this paper).

According to table 2, the proposed RBF training algorithm in this paper, gives more accurate results than those of Simple-PSO and newrb algorithms both on training and test sets.

TABLE 2. COMPARISON OF EXPERIMENTAL RESULTS OF THE TRAINING ALGORITHMS

Datasets	CmPSO		Simple-PSO		newrb	
	Train	Test	Train	Test	Train	Test
Iris	0.9932	0.9821	0.99	0.98	0.9850	0.9560
Wine	0.9992	0.9774	1	0.9631	0.9375	0.6554
Glass	0.8301	0.7748	0.8042	0.6620	0.9850	0.6174
New-thyroid	0.9721	0.9560	0.9650	0.9444	0.9240	0.6204

V. CONCLUSION

This paper proposed a new PSO-based RBF network training algorithm, in which the spreads of the radial functions in the hidden units also were taken into consideration. For this purpose only the centers and the widths were encoded into the particles in order to be optimized using PSO and the weights of the connections from hidden units to the output layer were computed analytically using SVD. The experimental results on different benchmark classification problems from UCI repository revealed the effectiveness and efficiency of the proposed method. It seems that the results can be further optimized using other optimization techniques like simulated annealing or chaos search in collaboration with the proposed algorithm.

REFERENCES

[1] Kennedy, J., Eberhart, R.C., "Particle Swarm Optimization," Proceedings of IEEE In: International Conference on Neural Networks, Piscataway, NJ (1995) 1942-1948.

[2] Broomhead, D., Lowe, D., "Multivariable Functional Interpolation and Adaptive Networks," Complex Systems 2 (1988) 321-355.

[3] Moody, J., Darken, C., "Fast Learning Networks of Locally-Tuned Processing Units," Neural Computation 3 (1991) 579-588.

[4] Vogt, M., "Combination of Radial Basis Function Neural Networks with Optimized Learning Vector Quantization," IEEE International Conference on Neural Networks (1993) 1841-1846.

[5] Simon Haykin, "Neural Networks: a comprehensive foundation," Prentice-Hall, Inc., Upper Saddle River, New Jersey (1999).

[6] Liu, Y., Qing, Z., Shi, Z.W., "Training Radial Basis Function Network with Particle Swarms," ISNN04, Springer-Verlag Berlin Heidelberg(2004)317-322.

[7] Krzyzak, T.Linder and G. Lugosi, "Nonparametric classification using radial basis function nets and empirical risk minimization," Proc. 12th Int. Conf. on Pattern recognition, Jerusalem, Israel (1994)72-76.

[8] Blake, C., Keogh, E., Merz, C.J.: UCI Repository of Machine Learning Databases www.ics.uci.edu/ mlearn/MLRepository.html.