International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:11, 2007

Training Radial Basis Function Networks with Differential Evolution

Bing Yu , Xingshi He

***Abstract*—**In this paper, Differential Evolution (DE) algorithm, a new promising evolutionary algorithm, is proposed to train Radial Basis Function (RBF) network related to automatic configuration of network architecture. Classification tasks on data sets: Iris, Wine, New-thyroid, and Glass are conducted to measure the performance of neural networks. Compared with a standard RBF training algorithm in Matlab neural network toolbox, DE achieves more rational architecture for RBF networks. The resulting networks hence obtain strong generalization abilities.

***Keywords*—**differential evolution, neural network, Rbf function

I. Introduction[1]

RADIAL Basis Function (RBF) networks were introduced into the neural network literature by Broomhead and Lowe [1], which are motivated by observation on the local response in biologic neurons. Due to their better approximation capabilities, simpler network structures and faster learning algorithms, RBF networks have been widely applied in many science and engineering fields. RBF network is three layers feedback network, where each hidden unit implements a radial activation function and each output unit implements a weighted sum of hidden units' outputs. Its training procedure is usually divided into two stages: First, the centers and widths of the hidden layer are determined by clustering algorithms such as K-means [2], vector quantizations [3], decision trees [4], and self-organizing feature maps [5]. Second, the weights connecting the hidden layer with the output layer are determined by Singular Value Decomposition (SVD) or Least Mean Squared (LMS) algorithms. The problem of selecting the appropriate number of basis functions remains a critical issue for RBF networks. The number of basis functions controls the complexity and the generalization ability of RBF networks. RBF networks with too few basis functions cannot fit the training data adequately due to limited flexibility. On the other hand, those with too many basis functions yield poor generalization abilities since they are too flexible and fit the noise in the training data. The methods mentioned above require designers to fix the structure of networks in advance according to prior knowledge. However it is difficult for designers to achieve optimal architecture. Genetic algorithm, the most popular evolutionary algorithms, has been employed to automatically evolve the structure of neural network [6]. Training technique can be formulated as an optimization problem, which includes the network structure into a set of variables that are used to minimize the prediction error. Differential Evolution (DE) algorithm, an emerging evolutionary computation technique, was first introduced by Rainer Storn and Kenneth Price in 1997[7]. DE possesses similar attractive features of genetic algorithms such as independence from gradient information of the objective function, the ability to solve complex nonlinear high dimensional problems. Furthermore, they can achieve faster convergence speed and require fewer parameters to be adjusted. In this paper, DE algorithm is adopted to auto-configure the structure of RBF network and obtain the model parameters according to given input-output examples.

Bing Yu , the Department of Mathematics, Xian University of Engineering Science and Technology, Xian 710048,P.R.China(e-mail:yubingxa@hotmail.com)
Xingshi He, the Department of Mathematics, Xian University of Engineering Science and Technology, Xian 710048,P.R.China

II. RBF Training Algorithm Design

The architecture of RBF network consists of three layers that have entirely different roles. The input layer, a set of source nodes, connects the network to the environment. The second layer consists of a set basis function units that carry out a nonlinear transformation from the input space to the hidden space. Usually, nonlinear transformation is based on Gaussian function as follows:

$$z_i(x) = \exp(-\frac{\|x-u_i\|^2}{2\sigma_i^2}) \quad (1)$$

Where ||…|| represents Euclidean norm; u_i, σ_i and z_i are the center, the width and the output of the i-th hidden unit, respectively. The output layer, a set of summation units, supplies the response of the network.

In this paper, we formulated the training technique as an optimization problem and employed DE algorithm to resolve it.

DE algorithm is a population-based heuristic search procedure. Start with NP individuals as solution vectors randomly, use mutation, crossing and selection operation through encoding, and then get the best individual as the problem's answer. By experimentation, we recently noticed that DE has exceptional performance compared to other search heuristics in numerical optimization. Surprisingly, DE requires hardly any parameter tuning and works very reliably with excellent overall results over a wide set of benchmark and real-world problems, such as partitioned data clustering [8], parameter identification of system models in engineering [9] and flexible ligand docking in bioinformatics [10].

When adopting DE to train RBF neural networks, we must resolve two problems: encoding neural network architecture and designing fitness function.

1) Encoding Neural Network Architecture

The aim of training is to determine the number of hidden units, centers and widths of corresponding hidden units, and the weights that connect hidden units and output units. The choice of an efficient representation for network architecture is one of the most important issues in training. If we encode all these parameters into a individual, the length of the individual is too long and hence the search space is too large, which results in a very slow convergence rate. Since the performance of RBF networks mainly depends on the centers of hidden units, we just encode the centers into a individual for stochastic search. Then the widths and weights are determined by heuristic methods and analytic methods, respectively. DE shows a strong ability to deal with real-valued optimization problem. Moreover, centers are also real values. In order to fully exploit potential of DE, real-valued flags are employed, which indicate whether or not the corresponding hidden units are involved in networks. Therefore, the position of a individual is represented by concatenation of flags and centers of hidden units. Suppose the max number of hidden units is set to h_{max}, thus the structure of the individual includes two parts as follows:

Center existence array				Center vector array			
$Flag_1$	$Flag_2$	…	$Flag_{hmax}$	C_1	C_2	…	C_{hmax}

Fig. 1 Structure of an individual

$Flag_i$ indicates whether or not the i-th hidden unit is involved into the network. If Flagi > 0, the i-th hidden unit is included in the network. Otherwise the i-th hidden unit is removed from the network. In such a way, individual can be interpreted to networks with variable number of hidden units though they have a fixed length. Here, different individual correspond to networks with different number of hidden units.

2) Designing Fitness Function

The fitness function guides the evolution process. Here a small fitness denotes a good individual. RBF networks with good configuration should have less hidden units and high prediction accuracy. Therefore, fitness function takes into account two factors: mean squared error between network outputs and desired outputs (*MSE*), and the complexity of networks (*ComNN*).

$$\text{Fitness}=\text{MSE}+k*\text{ComNN} \qquad (2)$$

Where k is a constant which balances the impact between MSE and ComNN

Calculating MSE **.** After an individual is interpreted into a neural network according to Section 2.1, the number of hidden units and their centers are obtained. Then the width of the *i*-th hidden unit is determined by the following heuristic formula:

$$\sigma_i^2 = \frac{1}{h}\sum_{j=1}^{h} \| Center_i - Center_j \|^2 \qquad (3)$$

Where h is the number of hidden units that are involved in the network.

Once the centers and the widths are fixed, the task of determining weights reduces to solving a simple linear system. An analytical non-iterative Single Value Decomposition (SVD) method is adopted. The resulting network is measured on the training set where PN input-target vector pairs (patterns) are given. The *MSE* is calculated as the following:

$$\text{MSE}=\frac{1}{\text{PN}}\sum_{p=1}^{PN} \| t_p - o_p \|^2 \qquad (4)$$

Where t_p and o_p are the desired output and network output for pattern p respectively.

Calculating ComNN

$$\text{ComNN}=\frac{N_{hidden}}{N_{maxhidden}} \qquad (5)$$

Where N_{hidden} is the number of hidden units involved in networks; $N_{maxhidden}$ is predefined the max number of hidden units.

III. Implementation with DE

Firstly, we get NP individuals indicating the RBF network' structure randomly, the individuals have the form:

$$x_{i,G} = [x_{1,i,G}, x_{2,i,G}, \ldots x_{D,i,G}] \quad i = 1, 2, \ldots, NP$$

(Where G is the generation number and D is the problem's dimension)

In each generation, for individual x_i, we using mutation operation and get the vector v_i called donor vector through formula 6

$$v_i = x_{r1,G} + F*(x_{r2,G} - x_{r3,G}) \qquad (6)$$

(Where the mutation factor F is a constant from [0, 2] and the three vectors $x_{r1,G}$, $x_{r2,G}$ and $x_{r3,G}$ are selected randomly such that the indices i, r_1, r_2 and r_3 are distinct from $1, 2, \ldots, NP$)

Secondly, we use crossing operation and get the trail vector u_i through formula 7.

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & if\ \text{rand}_{j,i} \le CR \text{ or } j = I_{rand} \\ x_{j,i,G+1} & f\ \text{rand}_{j,i} > CR \text{ and } j \ne I_{rand} \end{cases} \qquad (7)$$
$$i=1,2,...,NP; j=1,2,...,D$$

$\text{rand}_{j,i} \sim U[0,1], I_{rand}$ is a random integer from [1, 2,…, D], CR is a user defined number between [0,1] which called crossing factor.

Lastly, we use selection operation and get the target vector $x_{i,G+1}$ is compared with the trail vector $u_{i,G+1}$ and the one with the lowest function value is admitted to the next generation through formula 8. Mutation, crossing and selection continue until some stopping criterion is reached.

$$x_{i,G+1} = \begin{cases} u_{i,G+1} & if\ f(u_{i,G+1}) \le f(x_{i,G}) \\ x_{i,G} & \text{otherwise} \end{cases} \quad i = 1, 2, ..., NP \qquad (8)$$

The pseudo-code is as follows:

Begin
G=1;
Initialize the NP individuals $X_{i,G} : i = 1, 2, ..., N$ randomly;
For G=1 to Gmax do
For i =1 to NP do
Mutation step: for each individual $X_{i,G}$, get the donor vector $V_{i,G+1}$ according to formula 6;
Crossing step: get the trail vector $U_{i,G+1}$ according to formula 7 by vector $X_{i,G}$ and $V_{i,G+1}$;
Compute $J_{i,X}, J_{i,U}$ according to formula 2 by vector $X_{i,G}$, $U_{i,G+1}$ and train datasets;
Selection step: get the target vector $X_{i,G+1}$ according to formula 8;
End for
G=G+1;
End for
End

IV. Experiments

The data sets used in this section were obtained from the UCI repository of Machine Learning databases [11]. Two training algorithm were compared. One was DE algorithm whose parameters were set as follows: mutation factor F=0.3, crossing factor CR=0.7, Max iterations were different with different data sets. The other was newrb routine that was included in Matlab neural networks toolbox as standard training algorithm for RBF neural network. The function newrb iteratively creates a radial basis network one neuron at a time. Neurons are added to the network until the sum-squared error falls beneath an error goal or the maximum number of neurons has been reached. All the experiments were conducted 20 runs. In each experiment, each data set was randomly divided into two parts: 60% as training sets and 40% as test sets. TestCorrect referred to mean correct classification rate averaged over 20 runs for the test sets, respectively. The information of data sets and results of the two algorithms were listed in Table 1.

V. Conclusions

In this paper, DE algorithm, a population-based iterative global optimization, was implemented to train RBF networks. The method of encoding a RBF network into an individual was given, where only the centers of hidden units were encoded. In each iteration, each individual determined a kind of configuration for the centers of hidden units, according to which the widths of hidden units were calculated by heuristic methods, and connection weights between hidden layer and output layer were obtained by SVD. Consequently, a RBF

network was constructed. Then it was performed on training sets to evaluate fitness for the individual. Fitness function takes into account not only *MSE* between network outputs and desired outputs, but also the number of hidden units, thus the resulting networks can alleviate over-fitting. Experimental results show that DE algorithm achieves more rational architecture for RBF networks and the resulting networks hence obtain strong generalization abilities at the cost of a little longer time to train networks.

TABLE I
THE INFORMATION OF DATA SETS, PARAMETERS AND RESULTS OF THE TWO ALGORITHMS FOR DIFFERENT DATASETS

	Iris	Wine	New-thyroid	Glass
# of patterns	150	178	215	214
# of input units	4	13	5	9
# of output units	3	3	3	7
Max Iteration (DE)	100	150	200	200
# of hidden (DE)	6	20	15	27
Test Correct (DE)	0.9733	0.9631	0.9444	0.8620
# of hidden (newrb)	9	58	26	87
Test Correct (newrb)	0.7937	0.9283	0.5188	0.7857

REFERENCES

[1] Matysiak, Broomhead, D., Lowe, D.: Multivariable Functional Interpolation and Adaptive Networks. Complex Systems (1988) 321-355.
[2] Moody, J., Darken, C.: Fast Learning Networks of Locally-Tuned Processing Units. Neural Computation (1991) 579-588.
[3] Vogt, M.: Combination of Radial Basis Function Neural Networks with Optimized Learning Vector Quantization. IEEE International Conference on Neural Networks (1993) 1841-1846.
[4] Kubat, M.: Decision Trees Can Initialize Radial-Basis Function Networks. IEEE Transactions on Neural Networks (1998) 813-821.
[5] Robert, J., Hewlett L.C.J.: Radial Basis Function Networks 2: New Advances in Design (2001).
[6] Yao, X.: Evolving Artificial Neural Networks. Proceedings of the IEEE (1999) 87(9) 1423-1447.
[7] Rainer Storn, Kenneth Price: Differential Evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. *Global Optimization*, 11, 1997 341-359.
[8] PATERLINI, S., AND KRINK, T. High performance clustering using differential evolution. In Proceedings of the Six Congress on Evolutionary Computation (CEC-2004), IEEE Press, Piscataway, NJ, USA.
[9] THOMSEN, R. Flexible ligand docking using differential evolution. In Proceedings of the Fifth Congress on Evolutionary Computation (CEC-2003) (2003), vol. 4, IEEE Press, Piscataway, NJ, USA, pp. 2354–2361.
[10] URSEM, R. K., AND VADSTRUP, P. Parameter identification of induction motors using differential evolution. In Proceedings of the Fifth Congress on Evolutionary Computation (CEC-2003) (2003), IEEE Press, Piscataway, NJ, USA, pp. 790–796.
[11] Blake, C.,Keogh,E.,Merz,C.J.:UCI Repository of Machine Learning Databases(1998)www.ics.uci.edu/mlearn/MLRepository.html