ISSN 1 746-7233, England, UK
World Journal of Modelling and Simulation
Vol. 7 (2011) No. 4, pp. 307-311

Width optimization of Gaussian function by genetic algorithm in RBF networks

A. Golbabai* , A. Safdari-Vaighani

School of Mathematics, Iran University of Science and Technology, Narmak, Tehran 1684613114, Iran

(*Received February 6 2010, Accepted May 1 2011*)

Abstract. Conventionally, in radial basis function (RBF) network width factor is constructed by obtaining r-nearest neighbor rule or taking equal to a constant for all Gaussian functions. This paper proposes an approach for the construction of width factor using genetic algorithm to optimize the Gaussian function. Our experimental results show that our proposed optimal-based width outperforms the conventional RBFs. Hence, the width optimization is expected to be a new alternative way to the construction of RBF networks.

Keywords: genetic algorithms, radial basis function networks, width factor

1 Introduction

The artificial neural networks (ANNs) have been effectively applied in many areas[4, 6, 7, 11], such as function approximation, pattern recognition, signal processing, and time series prediction, ect., due to their strong learning capability. Among different kinds of ANNs, the radial basis function (RBF) networks are widely used. The RBF network is a three-layer feed-forward network that generally uses a linear transfer function for the output units and a nonlinear transfer function (normally the Gaussian function) for the hidden units.

In spite of a number of advantages if compared with other types of ANNs, such as better approximation capabilities, simple network structures and faster learning algorithms and the development of RBF networks still involves difficulties in selecting the network structure (the number of nodes in the hidden layers, i.e., the number of centers) and calculating the model parameters (e.g., centers, widths and weights). In the literature, several algorithms are proposed for the computation of centers[9]. However, very few papers are dedicated to width optimization of the Gaussian functions [1, 10, 12].

In this paper, our goal is computing the width of the Gaussian functions. For this aim we use GAs to determine width scaling factor and compute the standard deviation of the distance between the data and their corresponding centers. This process is more efficient, as we allow an optimal overlapping of the Gaussian functions.

In Section 2 of this paper we review RBF networks and their training algorithm. Framework of GAs is explained in Section 3. In Section 4 we explain width factor and effects caused by using width scaling factor. Experimental results of the proposed criterion and its compression with other methods are presented in Section 5.

2 Radial basis function networks

RBF networks are a major class of neural network model, where the distance between the input vector and a prototype vector determines the activation of a hidden unit. RBF methods become a popular technique

* Corresponding author. Tel.: +9821-73913426. *E-mail address*: golbabai@iust.ac.ir.

in the mid 1960s for performing exact interpolation of a set of data point in a high-dimensional space[13]. The basic technique provides an interpolation function which passes through every data point: Consider a mapping from a d-dimensional input space to a one-dimensional target space y, where the data set consists of N input vectors x_i, with corresponding targets y_i, $i = 1, \cdots, N$. A RBF neural network model[3, 12] can be obtained as follows: First, the number, M, of basis function is usually much less than the number, N, of data points. Second, the centers of the basis function no longer need to be given by input data vectors, whose value is determined in the training process. Third, each basis function can have its own width factor, σ_j, and appropriate widths can alternatively be determined during the training process. Finally we obtain the following form for the RBF neural network mapping.

$$\bar{y} = \sum_{j=1}^{M} w_j \phi_j(x) + b \,, \tag{1}$$

where the scaling factor w_j in Eq. (1) represents a weight that connects hidden node to the output node of the network. The constant term b in equation Eq. (1) represents a bias.

Several forms of basis function have been considered in previous research on RBF models, the most common being the Gaussian:

$$\phi_j(x) = \exp\left(-\frac{||x - c_j||^2}{2\sigma_j^2}\right), \tag{2}$$

where x is the d-dimensional input vector with elements x_i, c_j is the center, and σ_j is the width factor of basis function ϕ_j Eq. (2). The training algorithm consist of finding the parameters c_j, σ_j and w_j, such that $\bar{y}$ fits the unknown function y as close as possible. After the best-fit function is calculated, the performance of the RBF network is estimated by computing an error criterion. Assuming that N training data (x_i, y_i), $i = 1, \cdots, N$ are available, the RBF network training problem can be formulated as an optimization problem, where the root mean square errors (RMSE) between the real outputs and the network predictions $\bar{y}_i$ minimized with respect to the hidden node center locations and width factor σ_j, $j = 1, \cdots, M$,

$$\text{RMSE} = \left(\frac{1}{N}\sum_{j=1}^{N}(\bar{y}_j - y_j)^2\right)^{1/2}. \tag{3}$$

There is a number of papers that proposed several algorithms for the computing of centers, c_j[9] and the weights w_j[2]. A vector quantization scheme is employed to obtain center of the radial basis functions and weights are found by solving classical least squares problem. The minimum norm solution for $||y - \phi w||^2$ is the following form: $w = (\phi^T \phi)^{-1} \phi^T y$.

3 Framework of genetic algorithm (GA)

GA is an adaptive search procedure based on the mechanics of natural genetics and natural selection and has been used for a variety of search problems[8], GA starts by generating a random population. Each individual of the population represents a possible solution and is coded by a binary string (called "chromosome"). The "fitness", which is a measure of adaptation to environment, is calculated for each individual. After the evaluation of all the individuals, the next population is obtained by using genetic operations, the genetic operations used in GA are: Selection, Crossover and Mutation. As the algorithm proceeds, the individuals of the population are gradually improved. The operations used are explained as followers.

The chromosome with the lowest fitness has the lowest probability of selection, while the chromosome with the highest fitness has the greatest probability of selection. Assume that the population has g chromosomes, the probability of selection of $i-$th chromosome, p_i is given by $p_i = f_i / \sum_{i=1}^{g} f_i$, where f_i is the fitness value of $i-$th chromosome.

Crossover is the creation of one or more offspring from the parents selected in the selection process. The most common form of crossover involves two parents that produce two offspring. A crossover point is randomly selected between the first and last bits of the parent chromosomes. The parent chromosomes are split into two sections at this position. The two offspring chromosomes are formed by exchanging the second sections of the two parents. The probability p_c controls this operation.

Mutation is the second way by which GA explores a search space. A single point mutation changes 1 to 0, and vice versa. Mutation points are randomly selected from the total number of bits in the population. Increasing the number of mutations increases the algorithm's freedom to search outside the current region of variable space. The probability p_m controls this operation.

4 Optimal width

There are two alternatives to consider. In the first one we consider that all the Gaussian radial basis functions have the same standard deviation[10] as follows: $\sigma = d/\sqrt{2M}$, where M is the number of centers and d is the maximum distance between the chosen centers. Such a choice for the standard deviation σ dose not guarantee that the Gaussian function is not too peaked or too flat. In other word, this choice would be close the optimal solution when data uniformly were distributed in the input space that lead to centers distribute uniformly. Unfortunately in practice most real-life problems have non-uniform data distributions. Therefore, this method is not applicable so we use methods that their widths depend on the position of centers. In the second approach, width of each Gaussian function was estimated independently. This can be done by computing the standard deviation of the distance between the data and their corresponding centers. Moody and Darken[12] use the standard r-nearest-neighbor rule for computing of the width factors:

$$\sigma_j = \left(\frac{1}{r} \sum_{i=1}^{r} ||c_i - c_j||^2 \right)^{1/2}, \tag{4}$$

where the c_i, $i = 1, \cdots, r$, are the r-nearest neighbor of center c_j, commonly used $r = 2$ or 3. In practice such a choice of widths is better than previous case because this method offers the advantage of taking the distribution variations of the data into account. Even though, as we will show next, the widths values remain sub-optimal.

In our approach, let a cluster is a region associated to each center where we can compute the standard deviations σ_j^c of each data cluster in classical way, Eq. (4). Therefore we determine a width scaling factor q, for all Gaussian functions. The widths defined as: $\forall j, \sigma = q\sigma_j^c$.

The approximation function $\bar{y}$ is smoothed by doing this way, Since this property allow an optimal overlapping of Gaussian functions that in practice the process is more efficient. The optimal width scaling factor q depends on the function to approximating and depends on data distribution, but it is possible to compute the optimal factor q. For this, we apply genetic algorithm to obtain q corresponds to smallest root mean square error Eq. (3). The algorithm starts with an initial population of chromosomes, which represent factor q. The number of centers are defined on the input space. For each chromosome the weights between the hidden and the output layer are calculated and the objective function is computed. New generation are produced using a selection mechanism and other genetic operators. The algorithm stops after a specified number of generations have been completed. The chromosome that has produced the minimum value of the root mean square error is selected as the optimum neural network model.

Important steps for optimization widths and calculation of RBF networks are summarized as follows:
Step 1. Give input-output points for the training set;
Step 2. For the number of given cluster, find the clusters and centers;
Step 3. (a) For calculating scaling factor q, randomly create an initial population of individual character strings with a probability of 0.5 of each bit being 0 or 1. Set counter $g = 1$;
(b) Compute the objective function value (the root mean square error for the training data) for each chromosome;
(c) Compute the fitness value for each chromosome;

Step 4. Perform one-point crossover and mutation operation on the current population to generate new individuals in the search space. Set $g = g + 1$;
Step 5. If g is smaller than a number of generation, go to step 3. (b) otherwise go to step 6;
Step 6. The string that has given the optimum value is designated and obtain RBF neural network mapping.

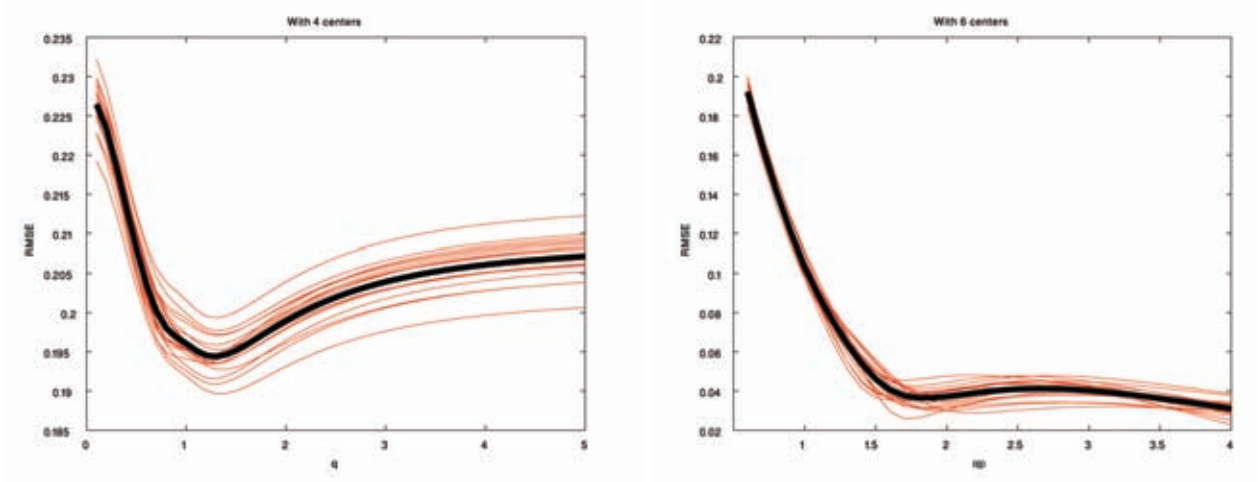

Fig. 1. RMSE curves and mean curve (thick line) against q with 6 centers for one-dimensional function and with 4 centers for two-dimensional function

5 Experiments and results

In this section we present numerical experiment to demonstrate the need of width optimization of the Gaussian functions in order to improve the generalization process. In our experiment the root mean squared error between the target and actual outputs is used as a performance measure.

5.1 Experiment conditions

In these experiments, the population size was chosen as 25 the crossover probability was 0.09 and the mutation probability was 0.05, the width scaling factor q was coded into an 8-bit binary string and the number of generation was set to 20. We run the algorithm 20 times, each time algorithm search optimum q to the interval $[0.5, 3]$. These results are obtained by repeatedly generating training samples of a fixed size and recording error achieved for each technique.

5.2 Result description

Two functions have been used to test our approach: The approximation of one-dimensional function, the approximation of two-dimensional function. Next, the characteristics of all of them presented.
(1) One-dimensional sine function: $f(x) = \sin^2 \pi x; \;\; 0 \leq x \leq 1$.
(2) Two-dimensional exponential function:

$$z = (1-x)^2 e^{(-x^2-(y+1)^2)} - 10\Big(\frac{x}{5} - x^3 - y^5\Big)e^{(-x^2-y^2)}; \;\; -3 \leq x, y \leq 3.$$

These example have been widely used in RBF neural network literature[5]. The obtained results by repeatedly generation of 500 data point for Example 1 and 1000 data point for Example 2 are reported and RMSE against q for these data points are shown in Fig. 1. The thick curve in Fig. 1 demonstrate mean of RMSE curves. In Tab. 1 and 2, we have compared our approach to the methods of Moody & Darken, S. Haykin. The numerical results presented in tables show the efficiency of proposed scheme in comparison with the exiting methods.

Table 1. Comparison of the test results of different methods for the one-dimensional sine function with 500 data point

Root Mean Square Error				
Centers	Our method	Moody & Darken	S. Hakin	Optimal q
5	0/0387	0/0428	0/0539	1/7415
6	0/0362	0/0393	0/0466	1/8760

Table 2. Comparison of the test results of different models for the two-dimensional exponential function with 1000 data point

Root Mean Square Error				
Centers	Our method	Moody & Darken	S. Hakin	Optimal q
4	0/1945	0/2052	0/2103	1/3115
6	0/1770	0/1871	0/1902	1/4560
8	0/1423	0/1482	0/1512	1/1941
10	0/1351	0/1420	0/1476	1/1016

6 Conclusion

This paper proposes an approach for the construction of width factor using genetic algorithm. In this method we let an optimal overlapping of Gaussian function to be created, which causes that this process to be more efficient practically. Finally, the results obtained by our approach based on GA in contrast with previous method is optimal.

References

[1] N. Benoudjit, C. Archambeau, et al. *Width optimization of the Gaussian kernels in RBF networks.* European Symposium on Artificial Neural Networks Bruges, 2002, 425–432.

[2] C. Bishop. *Neural Networks For Pattern Recognition*, Oxford University Press, Beijing, 1995.

[3] D. Broomhead, D. Lowe. Maltivariable functional interpolation and adaptive networks. *Complex Systems*, 2002, **2**: 321–1988.

[4] Y. Chena, B. Yanga, J. Dong. Time-series prediction using a local linear wavelet neural network. *Neurocomputing*, 2006, **69**: 449–65.

[5] A. Ghodsi, C. Schuurmans, Automatic basis selection techniques for RBF networks. *Neural Networks*, 2003. **16**: 806–816.

[6] C. Harpham, C. Dawson, M. Brown. A review of genetic algorithms applied to training radial basis function networks. *Neural Computing & Applications*, 2004, **13**: 193–201.

[7] A. Golbabai, S. Seifollahi. Radial basis function networks in the numerical solution of linear integro-differential equations. *Applied Mathematics and Computation*, 2007, **188**: 427–432.

[8] D. Goldberg. *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-Wesley, New York, Beijing, 1989.

[9] A. Gresho, R. Gary. *Vector Quantization and Signal Compression, International series in ingineering and computer science*, Kluwer Academic Publishers, Norwell, 1992.

[10] S. Haykin. *Neural networks: a comprehensive foundation*, Prentice-Hall, New Jersey, 1999.

[11] H. Mirzaee. Long-term prediction of chaotic time series with multi-step prediction horizons by a neural network with series with multi-step prediction horizons by a neural network with Levenberg-Marquardt learning algorithm. *Chaos, Solitons and Fractals*, 2009, **41**: 1975–1979.

[12] J. Moody, C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1989, **1**: 281–294.

[13] M. Powell. *Radial basis function for maltivariable interpolation: a review.* Algorithms for approximation (In J. Mason and M. Cox), Clarendon Press, Oxford, 1989, 143–167.