TIME SERIES PREDICTION USING WIDTH SCALING IN RBF NETWORKS

M. Moini¹, A. Golbabai² §

¹Department of Mathematics
Roudehen Branch
Islamic Azad University
Roudehen, IRAN

²Iran University of Science and Technology
Tehran, IRAN

Abstract: Conventionally, artificial neural network technique is used to predict the chaotic time series. This paper proposes an approach for the construction of width factor using genetic algorithm to optimize the Gaussian function in RBF networks. Our experimental results show that the developed evolving RBF networks are able to predict the chaotic time series.

AMS Subject Classification: 35F16
Key Words: genetic algorithms, Mackey-Glass time series, Radial basis function networks, width factor

1. Introduction

The artificial neural networks (ANNs) have been effectively applied in many areas [11], [1], [13], [14] such as function approximation, image processing, signal processing, and time series prediction, etc., 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 number advantages 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 [12], [5]. However, very few papers are dedicated to width optimization of the Gaussian functions [10], [7], [3].

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 in the mid 1960s for performing exact interpolation of a set of data point in a high-dimensional space [2]. 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 \( y_i \), \( i = 1, \ldots, N \). A RBF neural network model [4], [3] 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, \( \sigma_j \), and appropriate widths can alternatively be determined during the training process. Finally we obtain the following form
for the RBF neural network mapping.

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

(1)

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

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

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

(2)

where \( x \) is the \( d \)-dimensional input vector with elements \( x_i \), \( c_j \) is the center, and \( \sigma_j \) is the width factor of basis function \( \phi_j \). The training algorithm consist of finding the parameters \( c_j, \sigma_j \) 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 \((x_i, y_i), i = 1, \ldots, N\) 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 \( \sigma_j, j = 1, \ldots, M \),

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

(3)

There are a number of papers that several algorithms are proposed for the computing of centers, \( c_j \) [5] and the weights \( w_j \) [6]. 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 [9], 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 follows:

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

$$p_i = \frac{f_i}{\sum_{i=1}^{g} f_i}$$

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 [7] as follows:

$$\sigma = \frac{d}{\sqrt{2M}},$$
where $M$ is the number of centers and $d$ is the maximum distance between the chosen centers. Such a choice for the standard deviation $\sigma$ does 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 [3] use the standard $r$-nearest-neighbor rule for computing of the width factors:

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

where the $c_i$, $i = 1, \ldots, r$, 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 $\sigma_j^c$ of each data cluster in classical way. Therefore we determine a width scaling factor $q$, for all Gaussian functions. The widths defined as:

$$\forall j \quad \sigma = q\sigma_j^c.$$

The approximation function $\tilde{y}$ 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 approximate and depends on data distribution, so 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 (3).

Important steps for optimization widths and calculation of RBF networks to predict time series are summarized as follows:

(1) compute initial input-output points from Mackey-Glass by Runge-Kutta method.

(2) For the number of given cluster, find the clusters and centers.

(3) a) For calculating scaling factor $q$, randomly create an initial population of individual character strings. Set counter $g = 1$. 
b) Compute the root mean square error for each chromosome.

c) Compute the fitness value for each chromosome.

(4) Perform one-point crossover and mutation operation on to generate new individuals. Set $g = g + 1$.

(5) If $g$ is smaller than a number of generation, go to (3)-b otherwise go to 6.

(6) The string that has given the optimum value is designated and obtain RBF neural network mapping.

5. Experiments and Results

In this section we present some 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.

6. 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
7. Application to Mackey-Glass Time Series

The chaotic-Mackey Glass differential delay equation is recognized as a benchmark problem that has been used and reported by a number of researchers for comparing the learning and generalization ability of different neural architecture. The Mackey-Glass chaotic time series is generated from the following delay differential equation (see [11]):

$$\frac{dx(t)}{dt} = -bx(t) + a \frac{x(t - \tau)}{1 + x(t - \tau)^{16}}$$

The series has been generated using the next values for the parameters: $a = 0.2$, $b = 0.1$, and $\tau = 17$. The task for the RBF networks is to predict the value of the time series at point $x[t+1]$ from the earlier points $(x[t], x[t-6], x[t-12], x[t-18])$. Data points were generated with an initial condition $x(0) = 1.2$ based on the fourth-order Runge-Kutta method. Figure 1 shows the prediction results for Mackey-Glass time series and in table 1 we have prediction results using different number of nodes.

<table>
<thead>
<tr>
<th>Centers</th>
<th>Optimal(q)</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1/6111</td>
<td>0/0018</td>
</tr>
<tr>
<td>4</td>
<td>1/9245</td>
<td>0/0013</td>
</tr>
<tr>
<td>5</td>
<td>2/1423</td>
<td>0/0010</td>
</tr>
<tr>
<td>6</td>
<td>2/6825</td>
<td>0/0008</td>
</tr>
</tbody>
</table>

Table 1: Prediction results for Mackey-Glass timeseries using different number of nodes.

$q$ 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.

References


